



DB2 版本 9
适用于 Linux、UNIX 和 Windows



管理指南：计划

s151-0280-00





DB2 版本 9
适用于 **Linux**、**UNIX** 和 **Windows**



管理指南：计划

s151-0280-00

在使用本资料及其支持的产品之前，请务必阅读『*声明*』中的一般信息。

目录

第 6 章 设计分区数据库 189

第 7 章 针对符合 XA 的事务管理器进行设计 201

第 3 部分 附录 225

附录 A. 发行版之间的不兼容性 227

附录 B. 本地语言支持（NLS） 291

关于本书

《管理指南：计划》提供了使用和管理 DB2® 关系数据库管理系统（RDBMS）产品所需的信息，并包括有关数据库计划和设计的信息。

本书中描述的许多任务可以使用不同的界面来执行:

- **命令行处理器**，它允许您从命令行界面访问和处理数据库。从此界面，您也可执行 SQL 语句和 DB2 实用程序功能。本书中的大多数示例都演示了此界面的用法。有关使用命令行处理器的更多信息，请参阅 *Command Reference*。
- **应用程序编程接口**，它允许您在应用程序内执行 DB2 实用程序功能。有关使用应用程序编程接口的更多信息，请参阅 *Administrative API Reference*。
- **控制中心**，它允许您使用图形用户界面管理数据和数据库组件。您可以在 Linux™ 或 Windows® 命令行上使用`db2cc` 命令，或使用 Windows 平台的“开始”菜单来调用控制中心。控制中心以树形的对象层次结构呈现数据库组件。控制中心树包括系统、实例、数据库、表、视图、触发器和索引。通过该树，可以对数据库对象执行操作，例如：创建新表、重组数据、配置和调整数据库以及备份和恢复表空间。在许多情况下，提供有向导和启动板来帮助您快捷地执行这些任务。

  在下面三个视图中提供了“控制中心”：
  - 基本视图。此视图为您提供核心 DB2 数据库函数。您可以从该视图使用有权访问的所有数据库，包括其相关对象，如表和存储过程。它为您提供处理数据的基本要素。
  - 高级视图。此视图为您提供“控制中心”中的所有可用对象和操作。如果您在企业环境中工作，并且您想连接至 DB2 z/OS® 版（DB2 z/OS 版）或 IMS 版版本 9.1，则使用此视图。
  - 定制视图。此视图为您提供根据需要定制控制中心的功能。您可以选择想在视图中显示的对象和操作。

  要获取如何使用控制中心的帮助信息，从控制中心窗口的**帮助**下拉菜单中选择**入门**。

还可以使用其他工具来执行管理任务。这些工具包括:

- “命令编辑器”，它取代了“命令中心”，用来生成、编辑、运行和处理 SQL 语句; IMS 和 DB2 命令；使用生成的输出；以及查看说明 SQL 语句的访问方案的图形表示法。
- “开发中心”，它能够支持本机 SQL 持久存储器模块（PSM）存储过程，Java™ 存储过程 iSeries™ 版 V5R3 和更新版本，以及用户定义的函数（UDF）和结构类型。
- 运行状况中心，它提供了一个工具来帮助 DBA 解决性能和资源分配问题。
- “工具设置”，它用来更改“控制中心” 和“运行状况中心” 的设置。
- “内存可视化器”，它帮助数据库管理员监视实例的内存相关性能，以及以树形层次结构组织的所有实例的数据库。
- “不确定事务管理器”窗口，它用于显示不确定的事务。也就是在所选数据库和一个或多个所选分区中，正在等候落实、回滚或被遗忘的事务。

- “信息目录管理器”，它用于在仓库环境中工作时为数据关系和对象定义提供图形表示法。
- “日志”，它用来调度需要以无人照管方式运行的作业。
- “数据仓库中心”，它用来管理仓库对象。

本书面向的读者

本书主要面向需要策划和设计可以由本地或远程客户机访问的数据库的数据库管理员、系统管理员、安全性管理员和系统操作员。需要了解 DB2 关系数据库管理系统的管理和操作的程序员和其他用户也可使用本书。

本书的结构

本书各章节所讨论的主要主题区域如下:

数据库概念

- 第 1 章，『基本关系数据库概念』，提供数据库对象和数据库概念的概述。
- 第 3 章，『并行数据库系统』，提供有关 DB2 数据库可使用的并行性类型的介绍。

数据库设计

- 第 4 章，『逻辑数据库设计』，讨论逻辑数据库设计的概念和准则。
- 第 5 章，『物理数据库设计』，讨论物理数据库设计（包括空间需求和表空间设计）的准则。
- 第 6 章，『设计分区数据库』，讨论如何在单个事务中访问多个数据库。
- 第 7 章，『针对符合 XA 的事务管理器进行设计』，讨论如何在分布式事务处理环境中使用数据库。

附录

- 附录 A，『发行版之间的不兼容性』，描述版本 8 和版本 9 带来的不兼容性，以及未来计划的不兼容性。
- 附录 B，『本地语言支持（NLS）』，介绍“DB2 本地语言支持”，包括关于地域、语言和代码页的信息。

第 1 部分 数据库概念

第 1 章 基本关系数据库概念

关于数据库

关系数据库将数据表示成表的集合。表由一组已定义的列和任意数目的行组成。每个表中的数据在逻辑上是相关的，并且可以在表与表之间定义关系。可根据数学原理和关系运算（例如 INSERT、SELECT 和 UPDATE）来查看和处理数据。

除包含数据以外，数据库还包含其自身结构的描述，因此它具有自描述性。它包括一组系统目录表（这些表描述数据的逻辑和物理结构）；配置文件（它包含与数据库相关联的参数值）以及恢复日志（它记录正在进行的事务和可以归档的事务）。

数据库可以是本地的或远程的。本地数据库在物理上位于在使用的工作站上，而另一台机器上的数据库则被认为是远程的。

您可以:
- 创建数据库。
- 将数据库添加到控制中心中。
- 从控制中心中删除数据库。
- 备份数据库。
- 复原数据库。
- 配置数据库。
- 对数据库进行编目。
- 将数据库取消编目。
- 连接到数据库。
- 使用事件监视器来监视数据库。
- 使用分区数据库。
- 使用联合系统。

对于 z/OS 和 OS/390® 系统来说，缺省数据库 DSNDB04 是在 DB2 安装过程中定义的。此数据库具有缺省缓冲池（BP0）和缺省 DB2 存储器组（SYSDEFLT）。

相关概念:

- 『Tables』（*SQL Reference, Volume 1*）

数据库对象

系统:

DB2 数据库是围绕着数据库对象层次结构组织的。层次结构中的最高层对象是系统。系统表示安装的 DB2。控制中心维护它所了解的系统列表并记录与每个系统通信所需的信息（例如系统的网络地址、操作系统和通信协议）。控制中心既支持 DB2 系统也支持 IMS™ 系统。

系统可以有一个或多个 DB2 实例，每个实例可以管理一个或多个数据库。可以对数据库进行分区以使它们的表空间位于数据库分区组中。表空间存储表数据。

您可以:

- 将系统添加到控制中心中。
- 连接到系统。
- 从控制中心中除去系统。

实例:

实例（有时称为*数据库管理器*）是管理数据的 DB2 代码。它控制可对数据执行的操作，并管理分配给它的系统资源。每一个实例都是一个完整的环境。它包含为给定并行数据库系统定义的所有数据库分区。一个实例有它自己的数据库（其他实例不能访问它），并且它的全部数据库分区共享相同的系统目录。它还有独立的安全性，不受同一计算机（系统）上其他实例的影响。

数据库:

*关系数据库*将数据表示成表的集合。表由数目已定的列和任意数目的行组成。每个数据库都包括描述数据的逻辑和物理结构的一组系统目录表、包含为数据库分配的参数值的配置文件，以及带有运行事务和可归档事务的恢复日志。

数据库分区:

*数据库分区*由它自己的数据、索引、配置文件和事务键组成。它有时称为节点或数据库节点。表可以存储在一个或多个数据库分区中。当表数据分布在多个分区中时，它的某些行存储在一个分区中，而其他行存储在其他分区中。数据检索和更新请求被自动分解成子请求，并在适用的数据库分区中并行运行。数据库分布在各数据库分区中的这个事实对于用户是透明的。

数据库分区组:

*数据库分区组*是一个或多个数据库分区的集合。想要为数据库创建表时，首先创建用来存储表空间的数据库分区组，然后创建用来存储表的表空间。

在较早版本的 DB2 通用数据库™（UDB）中，*数据库分区组*称为*节点组*。

表空间:

数据库由称为*表空间*的部件组成。表空间是用来存储表的位置。当创建表时，您可以决定将特定对象（例如，索引和大对象（LOB））数据与其余表数据分开存放。表空间也可以分布在一个或多个物理存储设备上。下图表显示了在表空间之间分布数据时具有的一些灵活性。

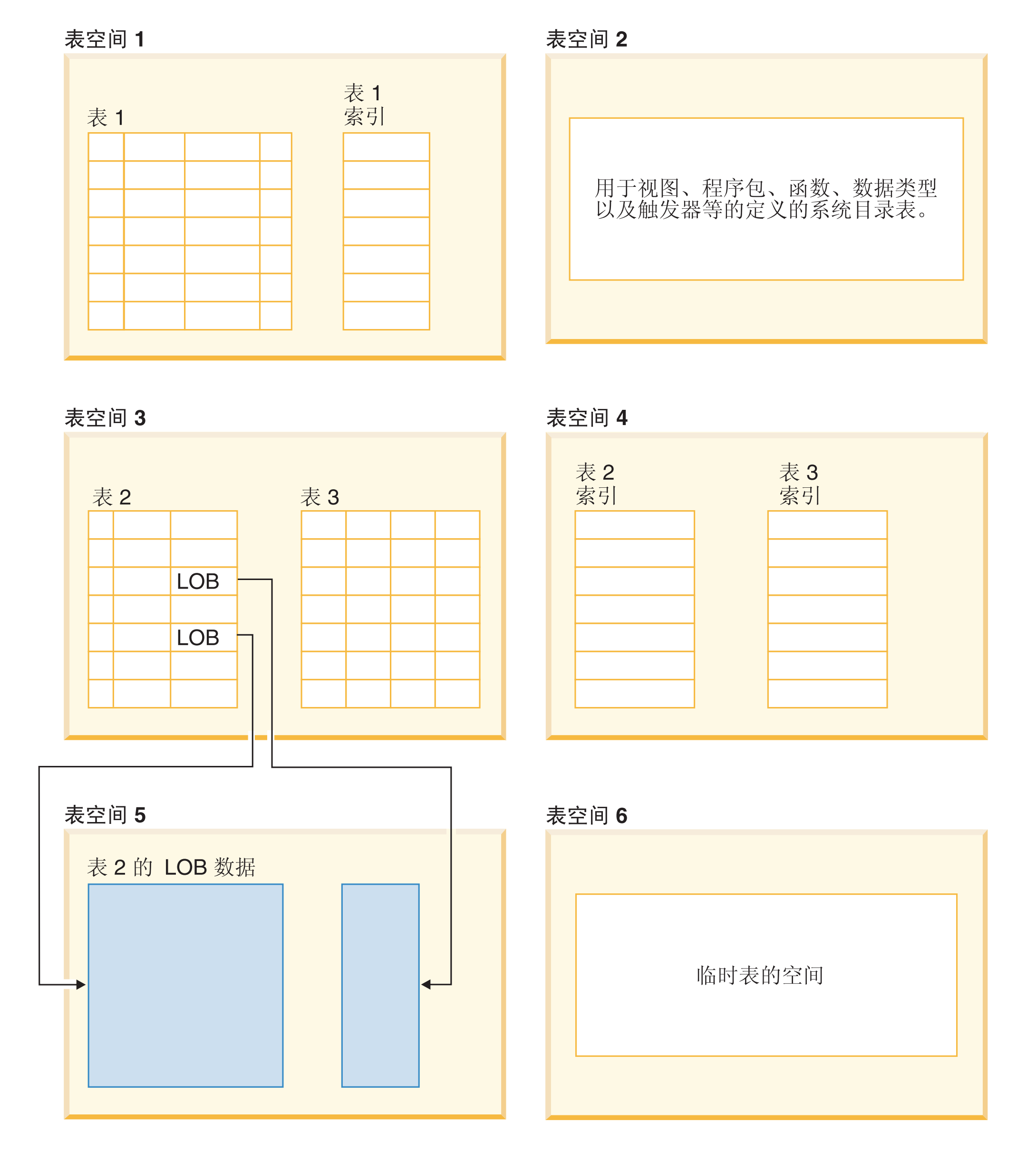


图 1. 表空间灵活性

表空间位于数据库分区组中。表空间定义和属性记录在数据库系统目录中。

将容器分配给表空间。容器是分配的物理存储器（例如，文件或设备）。

表空间可以是系统管理的空间（SMS）或数据库管理的空间（DMS）。对于 SMS 表空间，每个容器都是操作系统的文件空间中的一个目录，由操作系统的文件管理器控制存储空间。对于 DMS 表空间，每个容器或者是固定大小的预分配文件，或者是物理设备（例如，磁盘），由数据库管理器控制存储空间。

第 6 页的图 2 举例说明了表、表空间以及两种类型的空间之间的关系。它还显示了表、索引和长型数据存储在表空间中的情况。

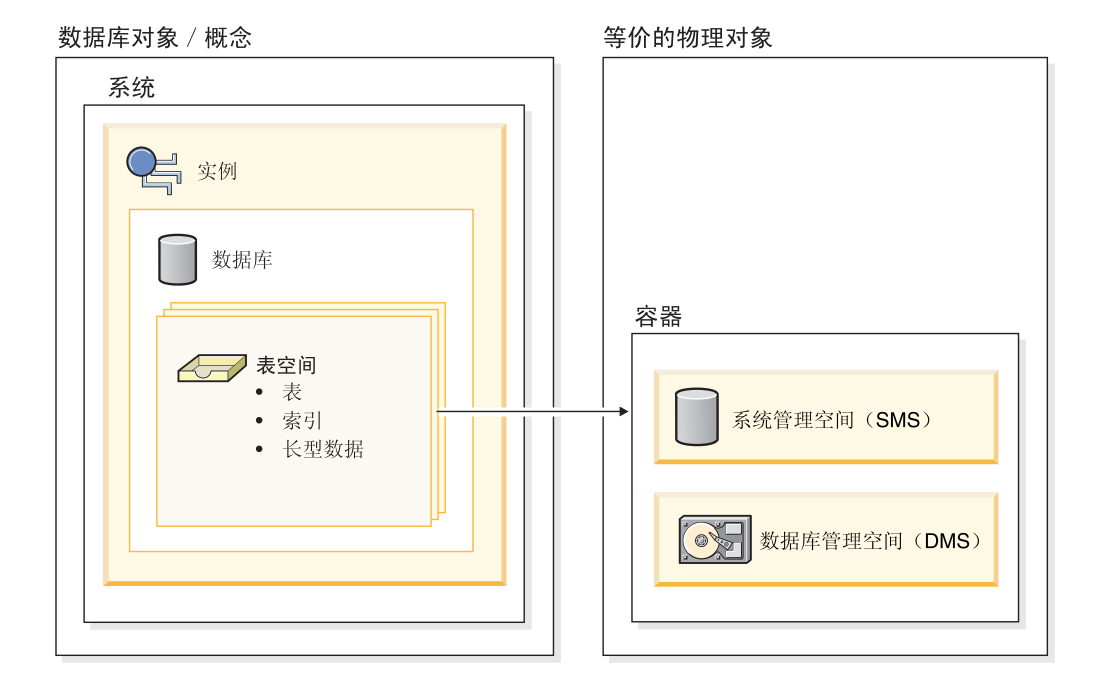


图 2. 保存数据的表空间和容器类型

第 7 页的图 3 显示了三种表空间类型：*常规*、*临时*和*大型*。

包含用户数据的表存放在常规表空间中。缺省用户表空间名为 USERSPACE1。系统目录表存放在常规表空间中。缺省系统目录表空间名为 SYSCATSPACE。

包含长型字段数据或大对象数据（例如，多媒体对象）的表存在于大型表空间或常规表空间中。这些列的基本列数据存储在常规表空间中，而长型字段或大对象数据可以存储在相同的常规表空间或指定大型表空间中。

索引可以存储在常规表空间或大型表空间中。

*临时表空间*分为系统临时表空间或用户临时表空间。*系统临时表空间*用来存储 SQL 操作（例如，排序、重组表、创建索引和连接表）期间所需的内部临时数据。这些操作需要额外的空间来处理结果集。虽然可以创建任意数目个系统临时表空间，但建议您只使用大多数表所使用的页大小创建一个。缺省系统临时表空间名为 TEMPSPACE1。任何用户和应用程序都可以使用系统临时表空间。*用户临时表空间*用来存储已声明全局临时表（已声明全局临时表存储的是应用程序临时数据）。在用户临时表空间内使用的表是使用 DECLARE GLOBAL TEMPORARY TABLE 语句创建的。用户临时表空间不是在数据库创建时缺省创建的。控制对用户临时表空间的访问。记住使用 GRANT 语句授予对用户临时表的适当 USE 特权。

数据库

常规表空间

表：
• 存储用户数据

临时表空间

• 系统临时表空间
• 用户临时表空间

大型表空间（可选）

表：
• 存储多媒体对象或其它大型对象数据

图 3. 表空间的类型

表:

关系数据库将数据表示成表的集合。表由逻辑排列的行和列数据组成。所有数据库和表数据都被存储在表空间中。表中的数据在逻辑上是相关的，且可以定义表与表之间的关系。可根据数学规则和关系运算来查看和处理数据。

表数据是通过“结构化查询语言”（SQL）访问的，SQL 是一种标准化语言，用于在关系数据库中定义和处理数据。应用程序或用户使用*查询*从数据库中检索数据。查询使用 SQL 来创建如下格式的语句

```
SELECT <data_name> FROM <table_name>
```

视图:

*视图*是高效率的数据呈示方法（无需维护数据）。视图不是实际的表，不需要永久存储器。“虚拟表”是即创建即使用的。

视图可以包括它所基于的表中的所有或某些列或行。例如，可以在视图中连接一个部门表和一个职员表，以便可以列示特定部门中的所有职员。

第 8 页的图 4 显示了表与视图之间的关系。

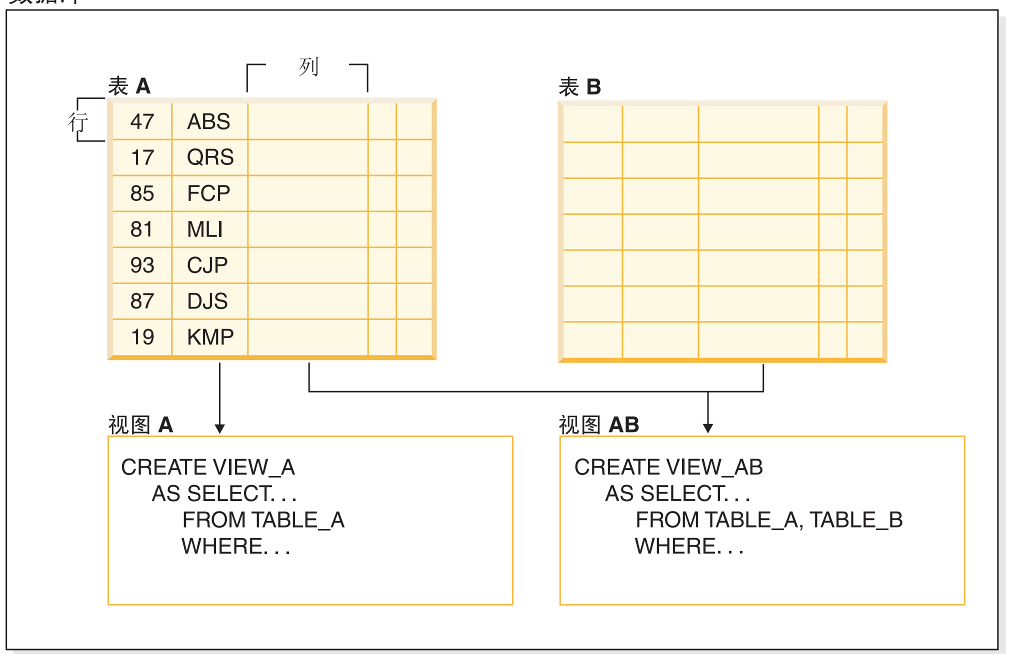


图 4. 表与视图之间的关系

索引:

将数据添加到表时，除非已对该表和正在添加的数据执行了其他操作，否则简单地将数据追加至该表的底部。数据是无序的。搜索特定数据行时，必须检查从第一行到最后一行的每个表行。将索引用作按顺序访问表中的数据的一种方法，该顺序在其他情况下可能不可用。

可以将一个数据行中的某个字段或列用作标识整个行的值。可能需要一列或多列来标识该行。这种标识列称为*键*。一列可以在多个键中使用。

索引按键中的值进行排序。

键可以是唯一的，也可以是非唯一的。每个表应该至少有一个唯一键；但还可以有其他非唯一键。每个索引正好有一个键。例如，可以使用职员标识编号（唯一）作为一个索引的键，并使用部门号（非唯一）作为另一个索引的键。

*索引*是一个或多个键的集合，每个键指向表中的一行。例如，在第 9 页的图 5 中，表 A 的一个索引基于表中的职员编号。此键值提供指向表行的指针。例如，职员编号 19 指向职员 KMP。索引允许通过指针创建指向数据的路径有效地访问表中的各行。

SQL *优化器*自动选择最有效率的访问表中数据的方法。当确定最快速的数据访问路径时，优化器会将索引考虑在内。

可创建唯一索引以确保索引键的唯一性。*索引键*是定义了索引的一个列或一些列的有序集合。使用唯一索引将确保在编入索引的列中，每个索引键的值都是唯一的。

第 9 页的图 5 显示了索引与表之间的关系。

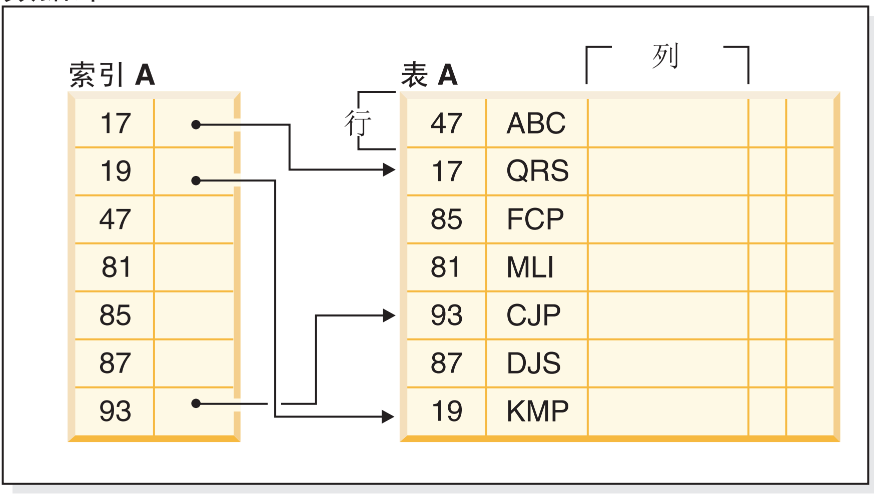


图 5. 索引与表之间的关系

图 6 说明一些数据库对象之间的关系。它还显示了表、索引和长型数据存储在表空间中的情况。



图 6. 所选择的数据库对象之间的关系

模式:

*模式*是一个标识，例如，用户标识，它帮助将表和其他数据库对象进行分组。模式可以被个别人所有，而所有者可以控制对模式中的数据和对象的访问。

模式也可以是数据库中的对象。它可以在创建模式中的第一个对象时自动创建。这样的对象可以是任何由模式名限定的对象，例如，表、索引、视图、程序包、单值类型、函数或触发器。如果要自动创建模式，则您必须拥有 IMPLICIT_SCHEMA 权限，也可以显式地创建模式。

模式名用作两部分对象名的第一部分。创建一个对象时，可将其分配给特定模式。若不指定模式，则它被分配给缺省模式，缺省模式通常是创建该对象的人员的用户标识。名称的第二部分是对象名。例如，名为 Smith 的用户可能具有名为 SMITH.PAYROLL 的表。

系统目录表:

每个数据库都包括一组描述数据的逻辑和物理结构的*系统目录表*。DB2 为每个数据库创建和维护许多系统目录表。这些表包含有关数据库对象（例如，用户表、视图和索引）的定义的信息，以及用户对这些对象所拥有的权限的安全性信息。这些信息是在创建数据库时创建的，并在常规操作期间得到更新。不能显式地创建或删除它们，但是可以使用目录视图查询和查看它们的内容。

容器:

*容器*是物理存储设备。可以用目录名、设备名或文件名来标识它。

将为表空间分配容器。单个表空间可以横跨多个容器，但每个容器只能属于一个表空间。

第 11 页的图 7 举例说明了表与数据库中的表空间、相关联的容器和磁盘之间的关系。

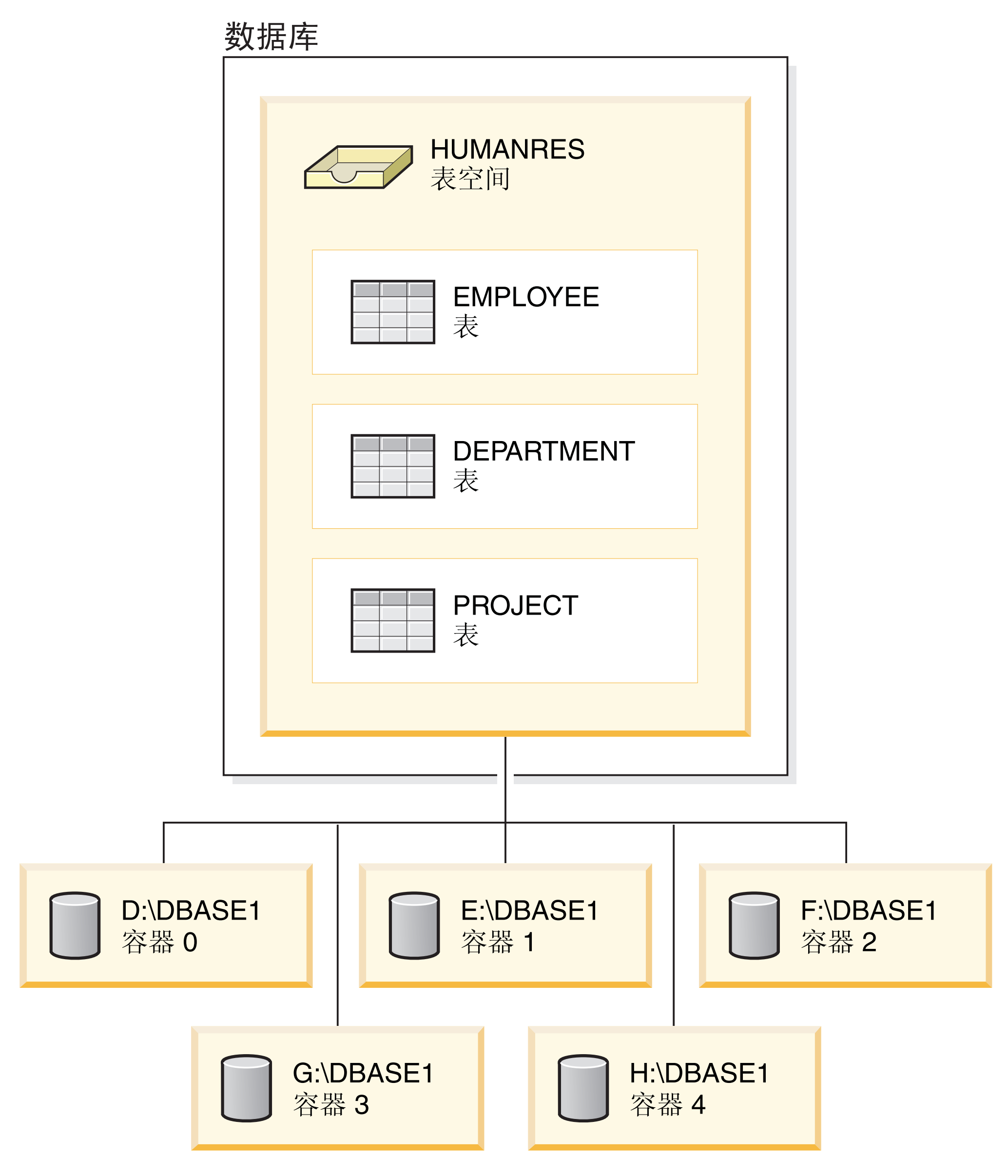


图 7. 表空间与它的容器之间的关系

EMPLOYEE、DEPARTMENT 和 PROJECT 表在 HUMANRES 表空间中，该表空间横跨容器 0、1、2、3 和 4。此示例显示每个容器存在于单独的磁盘中。

任何表的数据都以循环方式存储在表空间中的所有容器中。这能在属于给定表空间的容器之间平衡数据。数据库管理器在使用另一个容器之前写入一个容器的页数称为*扩展数据块大小*。

缓冲池:

*缓冲池*指的是从磁盘读取高速缓存表和索引数据页时或修改它们时分配给它们的主存储器。缓冲池的目的是改进系统性能。从内存访问数据要比从磁盘访问数据快得多；因此，数据库管理器需要读写磁盘（I/O）的次数越少，性能也越好。（可以创建多个缓冲池，虽然在大多数情况下只需要一个。）

因为可以缩短慢速 I/O 所造成的延迟，所以缓冲池的配置是最为重要的调整项目。

图 8 举例说明了缓冲池与容器之间的关系。

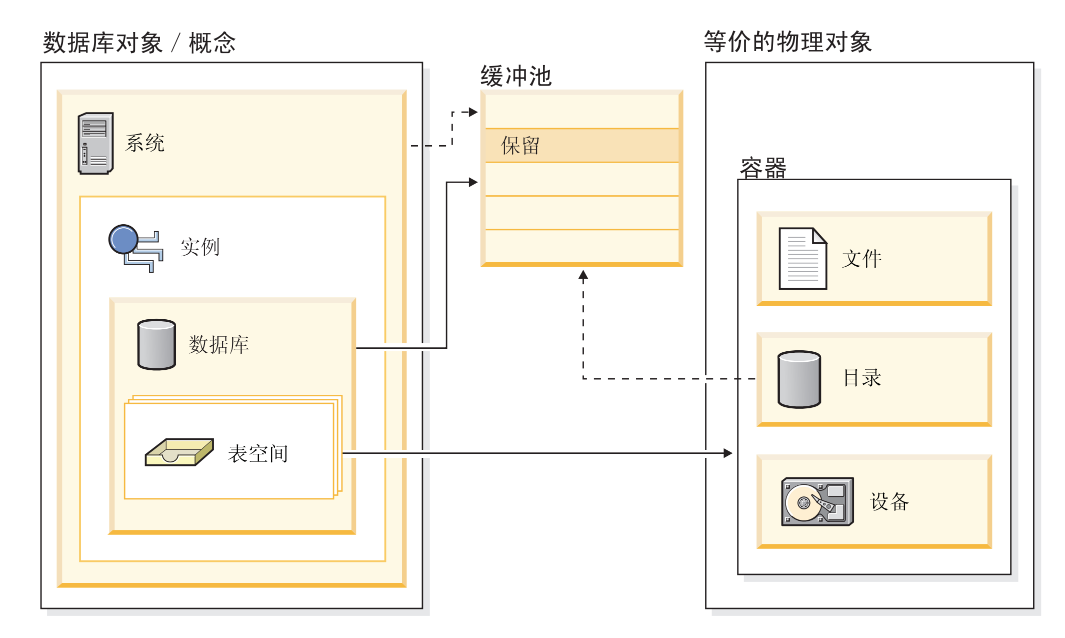


图 8. 缓冲池与容器之间的关系

相关概念:

- 『Indexes』 (*SQL Reference, Volume 1*)
- 『Relational databases』 (*SQL Reference, Volume 1*)
- 『Schemas』 (*SQL Reference, Volume 1*)
- 『Table spaces and other storage structures』 (*SQL Reference, Volume 1*)
- 『Tables』 (*SQL Reference, Volume 1*)
- 『Views』 (*SQL Reference, Volume 1*)

配置参数

当创建了 DB2 数据库实例或数据库时，就会使用缺省参数值创建相应的配置文件。可以修改这些参数值以提高性能以及实例或数据库的其他特征。

*配置文件*包含一些参数，这些参数定义诸如分配给 DB2 数据库产品和各个数据库的资源以及诊断级别之类的值。有两种类型的配置文件:

- 每个 DB2 实例的数据库管理器配置文件
- 每个独立的数据库的数据库配置文件。

*数据库管理器配置文件*是在创建 DB2 实例时创建的。它包含的参数在实例级影响系统资源，不受该实例包含的任何一个数据库影响。根据系统的配置，可将这些参数中许多参数的值更改为非系统缺省值，以提高性能或增大容量。

每个客户机安装也有一个数据库管理器配置文件。此文件包含关于特定工作站的客户机启用程序的信息。在可用于服务器的参数中，有一个子集可用于客户机。

数据库管理器配置参数存储在名为 `db2systm` 的文件中。此文件是在创建数据库管理器的实例时创建的。在基于 UNIX® 的环境中，可以在数据库管理器的实例的 `sqllib` 子

目录中找到此文件。在 Windows 中，此文件的缺省位置是 sqllib 目录的实例子目录。如果设置 DB2INSTPROF 变量，则文件在 DB2INSTPROF 变量指定的目录的 instance 子目录中。

在分区数据库环境中，此文件位于共享文件系统中，以便所有数据库分区服务器对同一文件都具有访问权。数据库管理器的配置在所有数据库分区服务器上都相同。

大多数参数会影响将分配给数据库管理器的单个实例的系统资源量，或者这些参数会配置数据库管理器和基于环境考虑的不同通信子系统的设置。另外，存在仅供参考的其他参数，不能更改它们。所有这些参数都具有全局适用性，独立于存储在数据库管理器的该实例下的任何单个数据库。

*数据库配置文件*是在创建数据库时创建的，它位于数据库所在的地方。每个数据库都有一个配置文件。其参数指定要分配给该数据库的资源量以及其他事项。您可以更改许多参数的值以提高性能或增大容量。根据特定数据库中活动类型的不同，可能需要进行不同的更改。

个别数据库的参数存储在名为 SQLDBCON 的配置文件中。此文件与 SQLnnnnn 目录中的数据库的其他控制文件存储在一起，其中 nnnnn 是创建数据库时指定的数字。每个数据库都有它自己的配置文件，并且文件中的大多数参数指定分配给该数据库的资源量。该文件还包含描述信息以及指示数据库的状态的标志。

在分区数据库环境中，对于每个数据库分区都存在一个单独的 SQLDBCON 文件。在每个数据库分区上，SQLDBCON 文件中的值可能相同或不同，但是建议数据库配置参数值在所有数据库分区上相同。

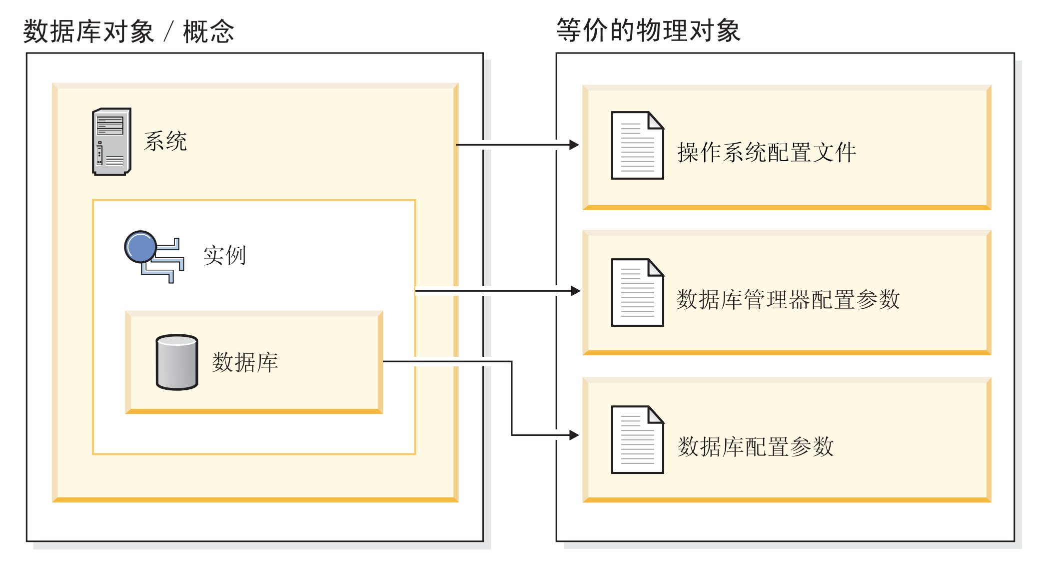


图 9. 数据库对象与配置文件之间的关系

相关概念:

- 『影响查询优化的配置参数』（*《性能指南》*）

相关任务:

- 『用配置参数配置 DB2』（*《性能指南》*）

环境变量和概要文件注册表

环境变量和注册表变量控制数据库环境。

可以使用“配置助手”（**db2ca**）来配置配置参数和注册表变量。

在引入 DB2 数据库概要文件注册表之前，在 Windows 工作站上更改环境变量（举例说明）要求您更改环境变量并重新启动。现在，除少数例外情况，您的环境均由 DB2 概要文件注册表中存储的注册表变量控制。在 UNIX 操作系统上，对于给定的实例来说，具有系统管理（SYSADM）权限的用户可更新该实例的注册表值。Windows 用户不需要 SYSADM 权限就可以更新注册表变量。使用 **db2set** 命令来更新注册表变量，而不需重新启动；此信息会立即存储在概要文件注册表中。DB2 注册表将已更新的信息应用于更改后启动的 DB2 服务器实例和 DB2 应用程序。

当更新注册表时，所作的更改不会影响当前正在运行的 DB2 应用程序或用户。在更新后启动的应用程序将使用新值。

注: 存在 DB2 环境变量 `DB2INSTANCE` 和 `DB2NODE`，它们可能未存储在 DB2 概要文件注册表中。在某些操作系统上，必须使用 **set** 才能更新这些环境变量。这些更改在下次重新启动系统之前一直有效。在 UNIX 平台上，可以使用 **export** 命令来代替 **set** 命令。

使用概要文件注册表允许集中控制环境变量。现在通过不同的概要文件提供了不同级别的支持。当使用 DB2 管理服务器时，还提供了对环境变量的远程管理。

有四个概要文件注册表:
- DB2 实例级概要文件注册表。大多数 DB2 环境变量都位于此注册表中。特定实例的环境变量设置保存在此注册表中。在此级别定义的值将覆盖在全局级的对应设置。
- DB2 全局级概要文件注册表。若未对特定的实例设置环境变量，则使用此注册表。对于属于 DB2 ESE 的特定副本的所有实例来说，此注册表可视；安装路径中存在一个全局级概要文件。
- DB2 实例节点级概要文件注册表。在分区数据库环境中，此注册表级别包含特定于数据库分区的变量设置。在此级别定义的值将覆盖实例级和全局级的对应设置。
- DB2 实例概要文件注册表。此注册表包含与当前副本相关联的所有实例名的列表。每个安装都有它自己的列表。可通过运行 `db2ilist` 来查看系统上提供的所有实例的完整列表。

DB2 通过按下列顺序检查注册表值和环境变量并解析它们来配置操作环境:
1. 使用 `set` 命令设置的环境变量。（或 UNIX 平台上的 `export` 命令。）
2. 使用实例节点级概要文件设置的注册表值（使用 **db2set -i <instance name> <nodenum>** 命令）。
3. 使用实例级概要文件设置的注册表值（使用 **db2set -i** 命令）。
4. 使用全局级概要文件设置的注册表值（使用 **db2set -g** 命令）。

实例级概要文件注册表

使用分区数据库环境时，UNIX 和 Windows 会有一些差异。这些差异如以下示例所示。

假设分区数据库环境有三个以“红色”、“白色”和“蓝色”标识的物理数据库分区。在 UNIX 平台上，如果实例所有者从任一数据库分区运行下述命令:

```
db2set -i FOO=BAR
```

或

```
db2set FOO=BAR        （“-i”是隐含的选项）
```

对于当前实例的所有节点（即“红色”、“白色”和“蓝色”节点），FOO 的值将是可视的。

在 UNIX 平台上，实例级概要文件注册表存储在 sqllib 目录下的文本文件中。在分区数据库环境下，sqllib 目录位于所有物理数据库分区共享的文件系统中。

在 Windows 平台上，如果用户从“红色”节点执行相同的命令，则 FOO 的值只对当前实例的“红色”节点可见。DB2 数据库管理器将实例级概要文件注册表存储在 Windows 注册表中。物理数据库分区之间没有共享。要设置所有物理计算机上的注册表变量，可使用以下“rah”命令:

```
rah db2set -i FOO=BAR
```

rah 将在“红色”、“白色”和“蓝色”节点远程运行 db2set 命令。

可以使用 DB2REMOTEPREG 将没有实例的计算机上的注册表变量配置为引用具有实例的计算机上的注册表变量。这样可以有效地创建以下环境：实例中的所有计算机共享具有实例的计算机上的注册表变量。

运用以上示例并假设“红色”节点为具有实例的计算机，即可通过下述操作将“白色”和“蓝色”计算机上的 DB2REMOTEPREG 设置为共享“红色”计算机上的注册表变量:

```
  （在红色机器上）不执行任何操作
（在白色和蓝色机器上）db2set DB2REMOTEPREG=\\red
```

DB2REMOTEPREG 的设置在设置后不必更改。

以下是 REMOTEPREG 的工作方法:

当 DB2 数据库管理器在 Windows 上读取注册表变量时，它首先读取 DB2REMOTEPREG 值。如果已经设置 DB2REMOTEPREG，则打开 DB2REMOTEPREG 变量中指定了名称的远程计算机上的注册表。将对指定的远程计算机重定向注册表变量的后续读取和更新。

访问远程注册表需要在目标计算机上运行“远程注册服务”。而且，用户登录帐户和所有 DB2 服务登录帐户对远程注册表有足够的访问权限。因此，要使用 DB2REMOTEPREG，需要在 Windows 域环境下操作，这样才能授予域帐户必需的注册表访问权限。

有一些 Microsoft® Cluster Server（MSCS）注意事项。不可在 MSCS 环境下使用 DB2REMOTEPREG。当属于同一 MSCS 集群的所有计算机运行在 MSCS 配置（在此配置下）中时，在集群注册表内维护注册表变量。因此，已在同一 MSCS 集群内的所有计算机之间共享它们，在此情况下没有必要使用 DB2REMOTEPREG。

当在多分区故障转移环境（在此环境中，数据库分区跨多个 MSCS 集群）中运行时，因为具有实例的计算机的注册表变量位于集群注册表中，所以不能用 DB2REMOTEPREG 来指向具有实例的计算机。

相关概念:

- 『DB2 注册表和环境变量』(《*性能指南*》)

相关任务:

- 『声明、显示、更改、注册和删除注册表和环境变量』(《*管理指南: 实施*》)

数据的业务规则

在任何业务中，数据通常必须符合特定限制或规则。例如，职员编号必须是唯一的。DB2 数据库 Linux 版、UNIX 版和 Windows 版提供了*约束*作为强制实施这种规则的方法。也可以使用触发器来对数据强制实施业务规则。

DB2 V9.1 提供了下列类型的约束:

- NOT NULL 约束
- 唯一约束
- 主键约束
- 外键约束
- 检查约束
- 参考约束

NOT NULL 约束

NOT NULL 约束防止将空值输入一个列。

唯一约束

唯一约束确保一组列中的值对于表中的所有行都是唯一的，且不为空。例如，DEPARTMENT 表中的典型唯一约束可以是：部门号是唯一的，且不为空。

下图显示了当表存在唯一约束时，阻止将重复的记录添加到该表。

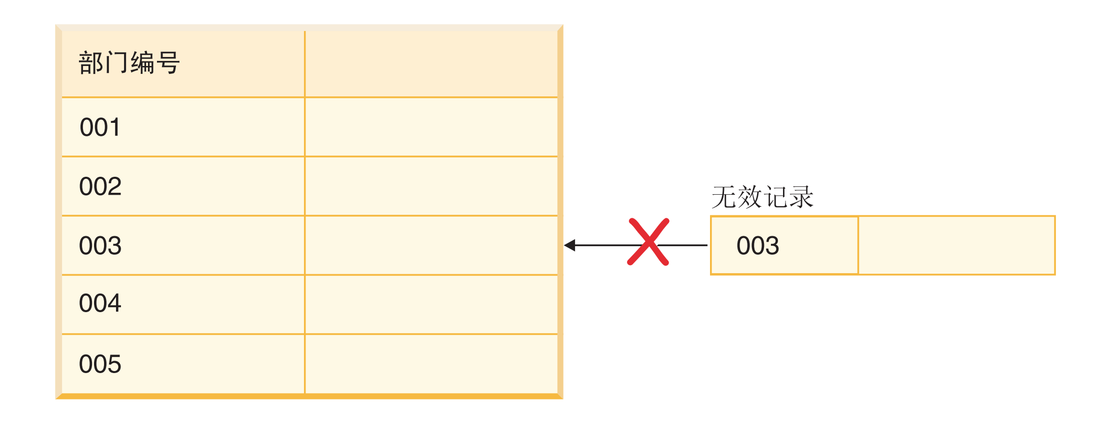


图 10. 唯一约束防止出现重复数据

数据库管理器在插入和更新操作期间强制执行此约束，以确保数据完整性。

主键约束

每个表都可以有一个主键。主键是与唯一约束具有相同属性的一个列或列的组合。可使用主键和外键约束来定义表之间的关系。

因为主键用来标识表中的一行，所以它应该是唯一的，并且只进行非常少的添加或删除。一个表不能有多个主键，但可以有多个唯一键。主键是可选的，可以在创建或改变表时定义。当导出或重组数据时，主键可以对数据进行排序，所以它们也是有益的。

在下面的表中，DEPTNO 和 EMPNO 是 DEPARTMENT 表和 EMPLOYEE 表的主键。

表 1. DEPARTMENT 表

DEPTNO（主键）	DEPTNAME	MGRNO
A00	Spiffy Computer Service Division	000010
B01	Planning	000020
C01	Information Center	000030
D11	Manufacturing Systems	000060

表 2. EMPLOYEE 表

EMPNO（主键）	FIRSTNAME	LASTNAME	WORKDEPT（外键）	PHONENO
000010	Christine	Haas	A00	3978
000030	Sally	Kwan	C01	4738
000060	Irving	Stern	D11	6423
000120	Sean	O'Connell	A00	2167
000140	Heather	Nicholls	C01	1793
000170	Masatoshi	Yoshimura	D11	2890

外键约束

外键约束（也称为引用完整性约束）使您能够定义表间以及表内必需的关系。

例如，典型的外键约束可能规定 EMPLOYEE 表中的每个职员必须是一个现有部门的成员，该部门在 DEPARTMENT 表中定义。

要建立此关系，应将 EMPLOYEE 表中的部门号定义成外键，并将 DEPARTMENT 表中的部门号定义成主键。

下图显示了当两个表之间存在外键约束时，如何阻止将具有无效键的记录添加到表。

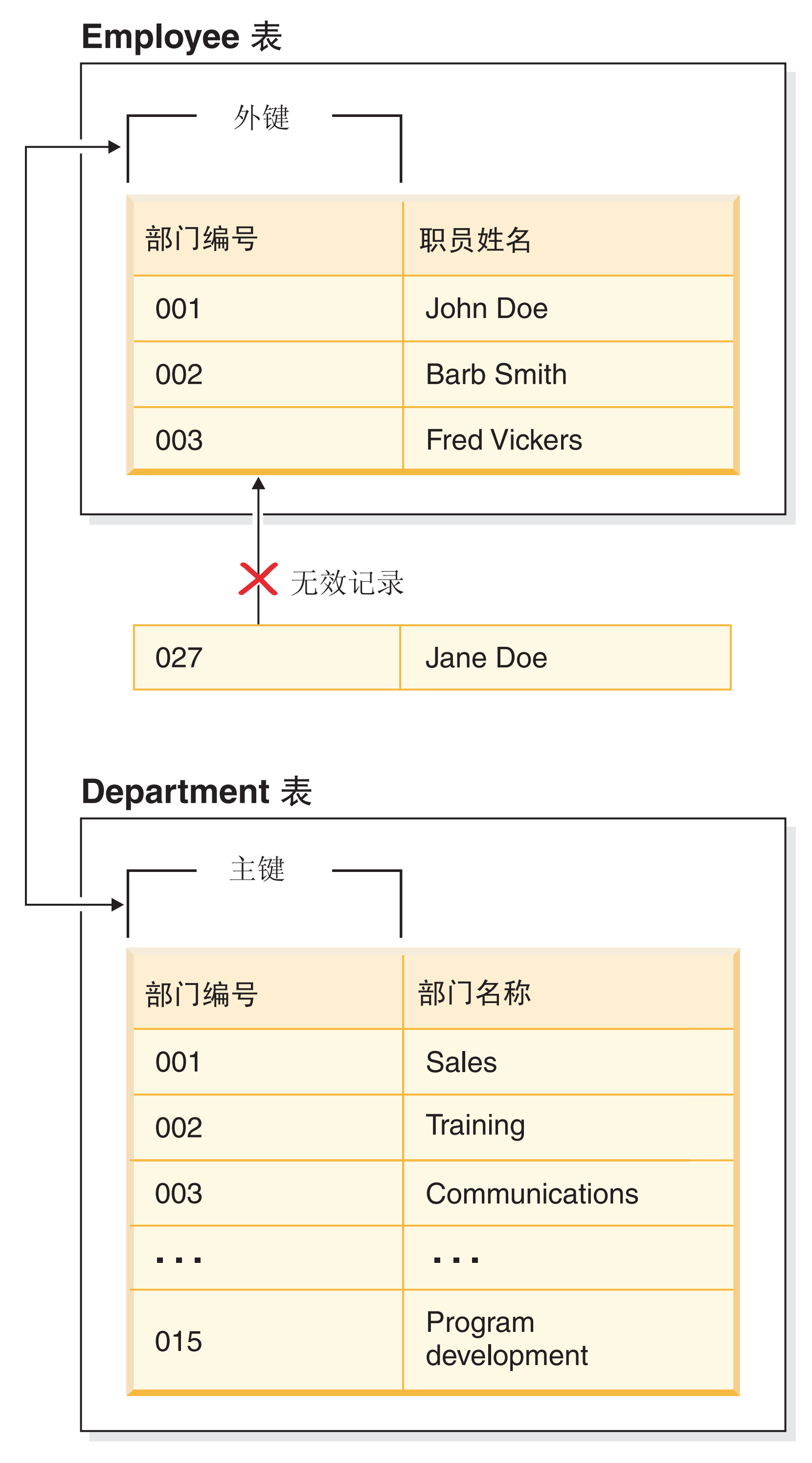


图 11. 外键和主键约束

检查约束

检查约束是指定每个表行的一列或多列所允许的值的数据库规则。

例如，在 EMPLOYEE 表中，可将“工作类型”列定义为“推销员”、“经理”或“职员”。有了此约束，在“工作类型”列中具有不同的值的任何记录都是无效的，将被拒绝，并实施关于表中允许的数据类型的规则。

参考约束

参考约束是 SQL 编译器可以使用的规则，但不是由数据库管理器强制执行的。该约束的目的不是由数据库管理器对数据执行附加验证，而是为了提高查询性能。

参考约束是使用 CREATE TABLE 或 ALTER TABLE 语句定义的。可以添加引用完整性或检查约束，然后使约束属性与它们相关联以指定数据库管理器是否强制执行约束；以及是否将约束用于查询优化。

除了使用约束对数据强制实施业务规则之外，还可以使用数据库中的*触发器*。与约束相比，触发器更为复杂，且潜在功能更加强大。它们定义一组操作，这组操作与用于指定基本表的 INSERT、UPDATE 或 DELETE 子句一起执行或由这些子句触发。可使用触发器支持一般形式的完整性或业务规则。例如，在接受订单之前，可使用触发器来检查客户的信用额度，也可以在银行业务应用程序中使用触发器，以便在帐户提款未符合客户的标准提款模式时发出警报。

相关概念:

- 第 60 页的『约束』
- 第 64 页的『触发器』

数据安全性

两个安全性级别控制着对 DB2 数据库 Linux 版、UNIX 版和 Windows 版数据和功能的访问。对 DB2 的访问由特定于操作环境的工具管理（认证），而 DB2 内的访问由数据库管理器管理（授权）。

认证就是系统验证用户身份的过程。用户认证是通过 DB2 外部的安全性工具（通常是操作系统部件或独立的产品）完成的。

一旦用户通过认证，数据库管理器就确定了是否允许该用户访问 DB2 数据或资源。授权是一个过程，DB2 通过此过程获取关于已认证用户的信息，从而指示该用户可以执行的数据库操作以及可以访问的数据对象。授权标识指定已授权用户的访问权。授权可以分为两类：特权和权限。

特权使用户能够创建或访问数据库资源。权限提供了对实例、数据库和数据库对象的特权分组方法，以及对其的维护和实用程序操作的控制方法。

相关概念:

- 第 3 页的『关于数据库』

认证

用户认证是通过使用 DB2 数据库 Linux 版、UNIX 版和 Windows 版之外的安全性工具来完成的。此安全性工具可以是操作系统的一部分，可以是一个独立的产品，或者，在特定情况下，可能根本不存在。在基于 UNIX 的系统上，安全性工具包含操作系统内部。

安全性工具需要两项来认证用户：用户标识和密码。用户标识向安全性工具标识用户。通过提供正确的密码（只有该用户和安全性工具才知道的信息）来验证该用户的身份（与该用户标识相对应）。

一旦认证:

- 必须使用 SQL 权限名或 *authid* 来向 DB2 标识该用户。此名称可以与用户标识或一个映射值相同。例如，在 UNIX 操作系统上，可将符合 DB2 命名约定的一个 UNIX 用户标识变换为大写字母来获得一个 DB2 *authid*。
- 获取用户所属的组的列表。在授权用户时，可使用组成员资格。组是安全性工具实体，它们也必须映射至 DB2 授权名。执行此映射的方法与映射用户标识的方法类似。

DB2 V9.1 通过下列两种方式之一，使用安全性工具来认证用户:

- DB2 使用成功的安全性系统登录作为身份证明，并允许:
  - 使用本地命令访问本地数据。
  - 当服务器信赖客户机认证时使用远程连接。
- DB2 接受用户标识和密码的组合。它使用安全性工具对此组合的成功验证作为身份证明，并允许:
  - 使用远程连接，在这种情况下服务器需要认证的证明。
  - 使用操作，在这种情况下用户希望以某个不同于注册时所用的标识来运行命令。

AIX® 上的 DB2 可将操作系统的失败密码尝试次数记入日志，并检测客户机何时超出允许的登录尝试次数，该值由 LOGINRETRIES 参数指定。

相关概念:

- 『服务器的认证方法』（*《管理指南：实施》*）
- 第 20 页的『授权』
- 『权限、特权和对象所有权』（*《管理指南：实施》*）

授权

授权是 DB2 获取有关已认证的 DB2 用户的信息的过程，此信息指示该用户可执行的数据库操作，以及可访问的数据对象。执行每个用户请求时，根据所涉及的对象和操作，可能有多个权限检查。

使用 DB2 工具来执行授权。DB2 表和配置文件用于记录与每个授权名相关的许可权。当一个已认证的用户尝试访问数据时，将该用户的授权名和他／她所属的那些组的授权名与记录的许可权进行比较。根据这个比较，DB2 决定是否允许执行请求的访问。

DB2 数据库 Linux 版、UNIX 版和 Windows 版记录了三种类型的许可权：特权、权限级别和 LBAC 凭证。

*特权*为授权名定义单个许可权，它使用户能够创建或访问数据库资源。特权存储在数据库目录中。

*权限级*提供将特权分组的一个方法，并控制更高级别的数据库管理器的维护和实用程序操作。特定于数据库的权限存储在数据库目录中；系统权限与组成员关系相关联，并且与权限级别相关联的组名存储在给定实例的数据库管理器配置文件中。

*LBAC 凭证*是 LBAC 安全标号和 LBAC 规则解除，它们允许访问受基于标号的访问控制（LBAC）保护的数据。LBAC 凭证存储在数据库目录中。

组提供了一个简便的方法来对一组用户执行授权，而不必单独对每个用户授予或撤销特权。除非另有指定，否则，组授权名可以用在为了授权而使用授权名的任何地方。通常，对于动态 SQL 和非数据库对象授权（如实例级命令和实用程序），考虑使用组成员资格，但对于静态 SQL，则不考虑使用它。在授予 PUBLIC 特权时则例外：在处理静态 SQL 时要考虑使用它们。DB2 文档中的适当地方提到了组成员关系不适用的特殊情况。

相关概念:

- 『Authorization and privileges』（*SQL Reference, Volume 1*）
- 『权限、特权和对象所有权』（*《管理指南：实施》*）
- 『基于标号的访问控制（LBAC）概述』（*《管理指南：实施》*）

工作单元

事务在 DB2 数据库 Linux 版、UNIX 版和 Windows 版中通常称为*工作单元*。工作单元是应用程序进程内可恢复的操作序列。数据库管理器使用它来确保数据库处于一致状态。对数据库的任何读取或写入都在一个工作单元内执行。

例如，银行事务可能涉及将存款从储蓄帐户转帐至支出帐户。当应用程序从储蓄帐户中扣除一定金额之后，这两个帐户将不一致，且这种不一致状况将在将该金额加入支出帐户之前一直保持。在两个步骤都完成后，便到达一致点。可以落实更改，并可使其用于其他应用程序。

当对数据库发出第一条 SQL 语句时，就隐式启动了一个工作单元。由同一应用程序执行的所有后续读写操作被认为是同一工作单元的一部分。应用程序必须发出 COMMIT 或 ROLLBACK 语句来结束该工作单元。COMMIT 语句使工作单元内的所有更改成为持久的。ROLLBACK 语句从数据库中除去这些更改。若应用程序正常结束，而未显式发出这两条语句中的任何一个，则自动落实该工作单元。若它在执行一个工作单元时异常结束，则自动回滚该工作单元。一旦发出 COMMIT 或 ROLLBACK，就不能停止它们。对于某些多线程应用程序或某些操作系统（例如，Windows），若应用程序正常结束，而未显式发出这两条语句中的任何一个，则自动回滚工作单元。建议应用程序始终显式地落实或回滚整个工作单元。若部分工作单元未成功完成，则回滚更新，使参与的表保持该事务开始前的状态。这确保了请求既不会丢失，也不会重复。

工作单元没有任何物理表示法，因为它是一系列指示信息（SQL 语句）。

相关参考:

- 『COMMIT statement』（*SQL Reference, Volume 2*）
- 『ROLLBACK statement』（*SQL Reference, Volume 2*）

高可用性灾难恢复（HADR）功能概述

DB2 数据库 Linux 版、UNIX 版和 Windows 版数据库高可用性灾难恢复（HADR）是一种数据库复制功能，它提供针对部分站点故障和整个站点故障的高可用性解决方案。HADR 通过将数据更改从源数据库（称为*主*数据库）复制至目标数据库（称为*备用*数据库）来防止丢失数据。

部分站点故障可能是由硬件、网络或软件（DB2 数据库或操作系统）故障引起的。如果不使用 HADR，就必须重新引导或重新启动数据库管理系统（DBMS）服务器或者数据库所在的机器。可能要花几分钟才能完成此过程。如果使用了 HADR，备用数据库就可以大约在几秒钟内接管主数据库角色。

当由于灾难（例如，火灾）而导致整个站点被破坏时，就可能会发生整个站点故障。由于 HADR 使用 TCP/IP 来进行主数据库与备用数据库之间的通信，所以这两个数据库可以位于不同的位置。例如，主数据库可能位于某个城市的总部，而备用数据库位于另一城市的销售办事处。如果在主要站点发生了灾难，则可以通过让远程备用数据库接管具有所有 DB2 功能的主数据库来维护数据可用性。在执行接管操作之后，可以通过对原始主数据库进行备份并使它恢复到主数据库状态；这就称为*故障回退*。

如果使用了 HADR，就可以通过指定下面三种同步方式中的任何一种方式来选择想要用于防止发生潜在的数据丢失的保护级别：同步（SYNC）、接近同步（NEARSYNC）和异步（ASYNC）。这三种方式指示如何在两个系统之间传播数据更改。选择的同步方式将确定备用数据库作为主数据库的副本的接近程度。例如，当使用同步方式时，HADR 可以保证在主数据库上落实的任何事务也会在备用数据库上落实。

同步允许您在两个系统之间实现故障转移和故障回退。

数据更改记录在数据库日志记录中，会将这些记录从主系统传递给备用系统。HADR 与 DB2 日志记录和恢复是紧耦合的。

HADR 要求两个系统具有相同的硬件、操作系统和 DB2 软件。（在这两个系统进行升级期间，它们之间可能存在一些细微的差别。）

HADR 备用数据库是通过将它从主数据库的备份复原或者通过从主数据库的分割镜像副本对它进行初始化来建立的。一旦启动了 HADR，备用数据库就会从主数据库检索日志记录并对主数据库其本身的副本重放这些记录。在备用数据库与主数据库的内存中的日志集“同步”之前，会将日志记录应用于备用数据库。此时，HADR 将把事务与 PEER 状态配对，在此期间，主数据库会将新的日志页发送至备用数据库，还会将这些日志页写入它的本地磁盘。当日志页到达时，在备用数据库上将重放日志页。通过连续不断地进行日志重放，备用数据库就成为主数据库的具有一定时间延迟的副本了。

当主数据库发生故障时，可以很容易地将它故障转移至备用数据库。一旦故障转移至备用数据库之后，它就成为新的主数据库。因为备用数据库服务器已处于联机状态，所以很快就可以完成故障转移。这就使没有数据库活动的时间减少到最小。

在进行某些硬件或软件发行版升级时，还可以使用 HADR 来维护数据库可用性。在对备用数据库升级硬件、操作系统或 DB2 修订包级别的同时主数据库仍可供应用程序使用。然后，在原始主数据库升级时，将这些应用程序传送至已升级的系统。

在故障转移之后，新的主数据库的性能可能不会立即与旧的主数据库在发生故障之前的性能完全相同。新的主数据库需要花一些时间来填充数据库管理器使用的语句高速

缓存、缓冲池和其他内存位置。尽管重放旧的主数据库的日志数据会将数据一部分放置在缓冲池和系统目录高速缓存中，但是，数据将是不完整的，这是因为它只是基于写活动。频繁访问的索引页、已查询但是未更新的表的目录信息、语句高速缓存和访问方案全部都将从高速缓存中丢失。但是，整个过程会比您在启动新的 DB2 数据库时更快。

一旦修复了发生故障的前一个主服务器，就可以将它重新集成为备用数据库（如果可以使该数据库的两个副本一致的话）。在重新集成之后，可以执行故障回退操作，以便使原始主数据库再次成为主数据库。

HADR 功能仅在 DB2 企业服务器版（ESE）上可用。在其他版本（例如，个人版）和具有数据库分区功能部件（DPF）的 ESE 中禁用该功能。

HADR 是在数据库级别而不是在实例级别进行的。这意味着单个实例可以包括主数据库（A）、备用数据库（B）和标准（非 HADR）数据库（C）。但是，一个实例不能同时包含单个数据库的主数据库和备用数据库，原因是 HADR 要求数据库的每个副本具有相同的数据库名称。

相关概念:

- 『高可用性』（*《数据恢复及高可用性指南与参考》*）

开发备份和恢复策略

数据库可能会因为硬件和／或软件故障而不可用。您可能会不时遇到存储问题、断电或应用程序故障以及各种需要采取不同恢复措施的故障情况。如果已准备了一个进行过彻底演习的恢复策略，它可保护您的数据免被丢失。在开发恢复策略时，您应该回答的一些问题包括:

- 数据库是可恢复的吗？
- 恢复数据库可能花费多长时间？
- 在备份操作之间将花费多长时间？
- 可以为备份副本和归档日志分配多少存储空间？
- 表空间级的备份是否足够？或是否需要进行完整的数据库备份？
- 是否应该通过手工或高可用性灾难恢复（HADR）来配置备用系统？

数据库恢复策略应确保在数据库恢复操作需要时，所有信息都可用。它应包括一个进行数据库备份的固定时间表，在分区数据库环境中则包括缩放系统时（添加或删除数据库分区服务器或节点时）的备份。完整的策略还应包括恢复命令脚本、应用程序、用户定义的函数（UDF）、操作系统库中的存储过程代码以及装入副本的过程。

在以下几节中还讨论了不同的恢复方法，您将能发现最适用于您的业务环境的恢复方法。

数据库*备份*的概念与其他任何数据备份的概念一样：即，复制一份数据，然后将它存储在另一介质上，以防原始介质发生故障或毁坏。最简单的备份情况需要关闭数据库（以确保不发生更多的事务），然后简单地对其进行备份。以后，如果数据库被损坏或毁坏，就可以重新创建它。

数据库的重新创建称为*恢复*。*版本恢复*指的是使用备份操作期间创建的映像来复原数据库的先前版本。*前滚恢复*是指复原了数据库或表空间备份映像后，重新应用记录在数据库日志文件中的事务。

*崩溃恢复*是指在完成并落实所有更改（这些更改是一个或多个工作单元（事务）的一部分）之前如果发生故障，会自动恢复数据库。这是通过回滚未完成的事务，并完成在发生崩溃时仍在内存中的已落实事务来实现的。

恢复日志文件和恢复历史记录文件是在创建数据库时自动创建的（图 12）。在需要恢复丢失或损坏的数据时，这些日志文件是很重要的。

每个数据库都包括*恢复日志*，它们用来从应用程序或系统错误中恢复。与数据库备份一起，它们用来将数据库的一致性恢复到出错时的时间点。

*恢复历史记录文件*包含当数据库的所有或部分必须恢复到给定时间点时，可用来确定恢复选项的备份信息的总结。恢复历史记录文件用来跟踪其他操作中与恢复相关的事件，如备份与复原操作。此文件位于数据库目录中。

表空间更改历史记录文件（它也位于数据库目录中）包含可用来确定哪些日志文件对特定表空间的恢复是必需的信息。

不能直接修改恢复历史记录文件或表空间更改历史记录文件；但是可以使用 PRUNE HISTORY 命令来删除文件中的条目。还可以使用 *rec_his_retentn* 数据库配置参数来指定这些历史记录文件将保留的天数。

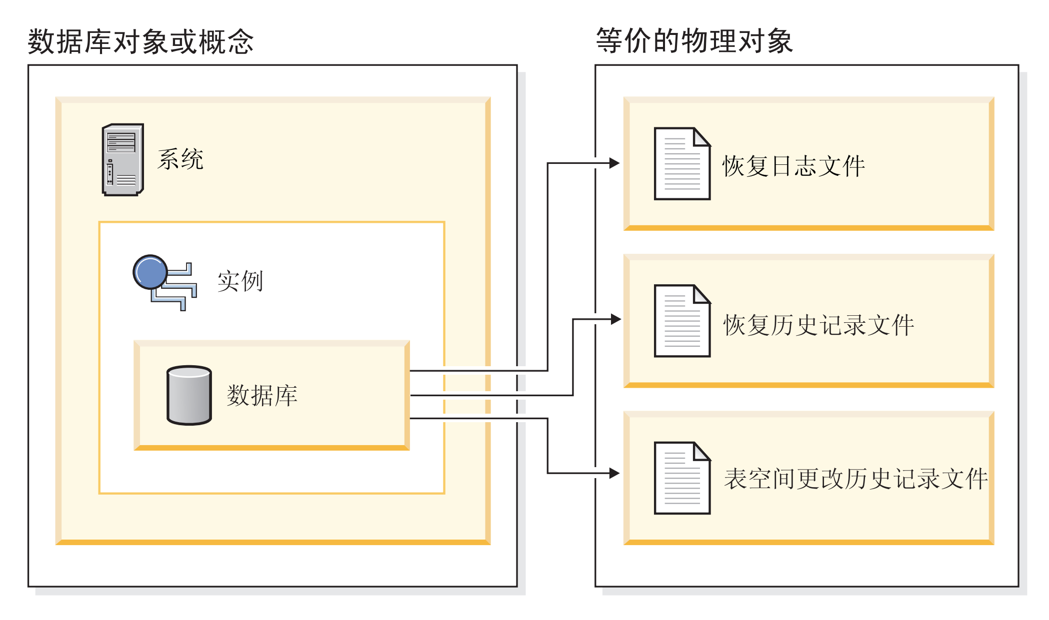


图 12. 数据库恢复文件

那些很容易重新创建的数据可存储在不可恢复数据库中。这些数据包括：用于只读应用程序的外部源中的数据以及不常进行更新的表；由于对它们进行的日志记录量较小，如果在复原操作之后还要进行复杂的日志文件文件管理工作和前滚操作，就不太合理了。如果 *logarchmeth1* 和 *logarchmeth2* 数据库配置参数都设置为 “OFF” ，则该数据库是*不可恢复的*。这表明将只保存崩溃恢复所必需的日志。这些日志称为*活动日志*，它们包含当前事务数据。使用*脱机*备份的版本恢复是解决不可恢复数据库的恢复

问题的主要手段。(脱机备份表示当备份操作正在进行时，其他应用程序无法使用该数据库。) 这样的数据库只能进行脱机复原。它被复原为生成备份映像时的状态且不支持前滚恢复。

那些不容易重新创建的数据应存储在可恢复的数据库中。这些数据包括在装入后它的源已被破坏的数据、手工输入到表中的数据以及在装入数据库后由应用程序或用户修改的数据。*可恢复数据库*将 *logarchmeth1* 或 *logarchmeth2* 数据库配置参数设置为“OFF”以外的值。活动日志仍可用于崩溃恢复，但您还有*已归档日志*，它包含已落实的事务数据。这样的数据库只能进行脱机复原。它被复原为创建备份映像时的状态。但对于前滚恢复，可通过使用活动日志和已归档日志来将数据库*前滚*(即，越过创建备份映像的时间) 至特定的时间点，或者滚动至活动日志末尾。

可恢复数据库备份操作可以脱机执行，也可以*联机*执行 (联机表示在备份操作期间其他应用程序可与该数据库连接)。仅当数据库可恢复时，才支持联机表空间复原和前滚操作。如果数据库是不可恢复的，必须脱机执行数据库复原和前滚操作。联机备份操作期间，前滚恢复确保捕获了*所有*表更改，且在复原该备份时重新应用这些更改。

若有一个可恢复数据库，则可备份、复原并将个别表空间前滚，而不必对整个数据库操作。当联机备份表空间时它仍然可用，同时发生的更新记录在日志中。当对表空间执行联机复原或前滚操作时，在该操作完成之前，该表空间本身不可用，但不阻止用户访问其他表空间中的表。

自动备份操作:

因为确定是否及何时运行维护活动 (例如，备份操作) 可能很费时间，所以可以使用“配置自动维护”向导来执行此操作。通过自动维护指定维护目标 (包括自动维护可以运行的时间)。然后，DB2 使用这些目标来确定是否需要执行维护活动，然后在下一可用维护窗口期间 (用户定义的、运行自动维护活动的时间段) 仅运行所需的维护活动。

注: 配置自动维护时，仍可执行手工备份操作。如果需要自动备份操作，DB2 将只执行这些自动备份操作。

相关概念:

- 『崩溃恢复』(*《数据恢复及高可用性指南与参考》*)
- 『高可用性灾难恢复概述』(*《数据恢复及高可用性指南与参考》*)
- 『前滚恢复』(*《数据恢复及高可用性指南与参考》*)
- 『版本恢复』(*《数据恢复及高可用性指南与参考》*)

相关参考:

- 『logarchmeth1 － 主日志归档方法配置参数』(*《性能指南》*)
- 『rec_his_retentn － 恢复历史记录保留期配置参数』(*《性能指南》*)

第 2 章 自动维护

关于自动维护功能

DB2 产品提供了自动维护功能，即执行数据库备份、保持统计信息是最新的以及在必要时重组表和索引。

对于确保数据库具有最佳性能和可恢复性来说，对数据库执行维护活动十分必要。这些活动包括:

- 备份数据库。DB2 将复制数据库数据并将其存储在另一介质上，以防原始介质发生故障或毁坏。自动数据库备份功能为用户提供了一个解决方案，从而帮助用户确保定期正确地对他们的数据库进行备份，而不必担心何时进行备份，也不需要具有有关备份命令的任何知识。
- 数据碎片整理（表或索引重组）。此维护活动可以提高 DB2 数据库管理器访问表的效率。自动重组功能管理脱机进行的表和索引重组，用户不必担心何时以及如何重组他们的数据。
- 数据访问优化（运行统计）。DB2 将更新有关表数据、表索引数据或者表数据及其索引数据的系统目录统计信息。优化器使用这些统计信息来确定用来访问数据的路径。自动收集统计信息功能通过维护最新的表统计信息来尝试提高数据库的性能。目标是允许优化器根据准确的统计信息来选择访问方案。
- 统计信息概要分析。自动统计信息概要分析功能通过检测过时的、丢失的以及不正确指定的统计信息以及通过根据查询反馈信息来生成统计信息概要文件的方法来建议何时以及如何收集表统计信息。

对于用户来说，确定是否以及何时运行维护活动可能相当费时。自动维护功能解除了用户的负担。借助自动维护功能，可以指定维护目标（包括自动维护功能的运行时间）。DB2 使用您指定的目标来确定是否需要执行维护活动，在下一个可用的维护窗口（自动维护活动的运行时间段，由用户定义），将仅运行必需的维护活动。

自动维护功能部件的启用是通过使用自动维护数据库配置参数来控制的。具有一组分层的开关用来方便灵活地管理这些功能部件的启用。可以使用“配置自动维护”向导使数据库维护活动仅在需要时才自动运行。DB2 数据库管理器使用已使用“配置自动维护”向导指定的目标来确定是否需要执行维护活动。然后，在下一个可用的维护窗口中，DB2 仅运行必需的维护活动。维护窗口是您定义的自动维护活动的运行时间段。

相关概念:

- 第 32 页的『维护窗口』
- 第 33 页的『脱机维护』
- 第 33 页的『联机维护』

缺省情况下启用的自动功能

DB2 包括了一些自动功能，缺省情况下在创建数据库时在会启用它们。这些功能用来帮助您管理数据库系统。这表示您的系统能够自诊断，并且能够在问题出现之前通过针对历史问题数据监视实时数据来预见问题。在某些情况下，可以配置这些工具以自动对系统进行更改，从而避免出现服务扰乱。

缺省情况下启用下列自动功能:

自动收集统计信息
: 自动收集统计信息通过收集最新的表统计信息来改善数据库性能。DB2 确定工作负载需要哪些统计信息以及需要更新哪些统计信息。然后，在后台自动调用 RUNSTATS 实用程序以确保收集并维护正确的统计信息。将首先收集有关最需要的表的统计信息。然后，DB2 优化器根据准确的统计信息来选择访问方案。在创建数据库之后，可以通过将数据库配置参数 AUTO_RUNSTATS 设置为 OFF 来禁用自动收集统计信息。

自动存储器
: 自动存储器功能简化了表空间的存储管理。创建数据库时，可以指定 DB2 将用来存放表空间数据的存储路径。然后，在创建和填充表空间时，DB2 将管理这些表空间的容器和空间分配。

配置顾问程序
: 创建数据库时，将自动调用此工具来确定并设置数据库配置参数和缺省缓冲池（IBMDEFAULTBP）的大小。根据系统资源和系统的用途选择值。此初始自动调整意味着数据库与使用 DB2 缺省值创建的数据库相比具有更好的性能。它还意味着在创建数据库之后，将使用较少的时间来调整系统。可以在任何时间调用“配置顾问程序”（即使在填充了数据库之后），以根据当前系统特征建议一组配置参数并且可以选择应用这些参数来优化 DB2 性能。

运行状况监视器
: “运行状况监视器”是一个服务器端工具，它主动监视数据库环境中可能导致性能下降或潜在中断的情况或变动。不需要您进行任何形式的监视活动就可以产生一些运行状况信息。如果遇到运行状况危险，DB2 将会让您知道这一点，并且还会建议您如何继续。“运行状况监视器”通过使用快照监视器来收集关于系统的信息，并且不会造成性能损失。此外，它不打开任何快照监视开关来收集信息。

自调整内存（仅适用于单一分区数据库）
: 此功能根据系统工作负载的内存需求来自动调整几个内存配置参数的值，从而简化了内存配置任务。内存调整器会在几个内存使用者（包括排序、程序包高速缓存、锁定列表和缓冲池）之间动态地分配可用内存资源。内存调整器对工作负载特征的显著更改作出响应，并不断调整内存配置参数值和缓冲池大小以优化性能。在创建数据库之后，可以通过将数据库配置参数 SELF_TUNING_MEM 设置为 OFF 来禁止对内存进行自调整。

实用程序调速
: 此功能调整维护实用程序的性能影响，以便可以在生产期间同时运行它们。虽然缺省情况下定义了已调速实用程序的影响策略，但在调用实用程序时必须设置影响优先级以便它调速运行。调速系统将确保已调速实用程序在不违反影响策略的情况下尽可能主动地运行。目前，可以调速统计信息收集、备份操作和重新平衡操作。

相关概念:

- 『运行状况监视器简介』(*《系统监视器指南和参考》*)
- 『自动存储器数据库』(*《管理指南: 实施》*)
- 『性能调整的快速启动技巧』(*《性能指南》*)
- 『自调整内存』(*《性能指南》*)
- 『自动收集统计信息』(*《性能指南》*)

相关参考:

- 『db2AutoConfig API - Access the Configuration Advisor』(*Administrative API Reference*)
- 『SET UTIL_IMPACT_PRIORITY command』(*Command Reference*)
- 『util_impact_lim – 实例影响策略配置参数』(*《性能指南》*)
- 『auto_maint – 自动维护配置参数』(*《性能指南》*)

自动数据库备份

由于大量发生硬件或软件故障，数据库可能会变得不可用。自动数据库备份功能减轻了 DBA 的数据库备份管理任务，它始终会确保在需要时对数据库执行最新的完全备份。它根据下面的一种或多种情况来确定是否需要执行备份操作:

- 您从未完成完整的数据库备份
- 自从上一次执行完全备份以来所经过的时间超过了指定的小时数
- 自从上一次执行备份以来所消耗的事务日志空间超过了指定的 4 KB 页数(仅适用于归档日志记录方式)。

通过为系统规划和实现灾难恢复策略来保护数据。如果能够满足您的需要，则可以将自动数据库备份功能部件作为您的备份和恢复策略的一部分。

如果对数据库启用了前滚恢复(归档日志记录)，则可以对联机备份或脱机备份启用自动数据库备份功能。否则，只能进行脱机备份。自动数据库备份支持磁盘、磁带、Tivoli® Storage Manager(TSM)和供应商 DLL 介质类型。

通过“控制中心”或“运行状况中心”中的“配置自动维护”向导，您可以配置:

- 请求备份之间的时间或日志页数
- 备份媒体
- 它将是联机备份还是脱机备份。

如果选择了备份至磁盘，则自动备份功能部件将定期从“配置自动维护”向导中指定的目录中删除备份映像。只能保证在任何给定时间都可以提供最新的备份映像。建议将此目录专用于自动备份功能部件，而不用来存储其他备份映像。

可以使用 **auto_db_backup** 和 **auto_maint** 数据库配置参数来启用或禁用自动数据库备份功能部件。在分区数据库环境中，如果对每个数据库分区都启用了数据库配置参数，则每个数据库分区上都将运行自动数据库备份。

相关概念:

- 第 23 页的『开发备份和恢复策略』

- 『自动收集统计信息』（《性能指南》）
- 『目录统计信息』（《性能指南》）
- 『MDC 表的表和索引管理』（《性能指南》）
- 『标准表的表和索引管理』（《性能指南》）
- 『表重新组织』（《性能指南》）
- 『运行状况监视器』（《系统监视器指南和参考》）

相关参考:

- 『auto_maint － 自动维护配置参数』（《性能指南》）

自动重组

在对表数据进行许多更改之后，逻辑上连续的数据可能会位于不连续的物理页面上，因此，数据库管理器必须执行附加读操作才能访问数据。

由 RUNSTATS 收集的统计信息与其他信息一起来显示表中的数据分发情况。特别是，通过分析这些统计信息可以知道何时需要执行哪种类型的重组。自动重组通过使用 REORGCHK 公式来确定何时需要对表进行重组。它会定期评估已经更新了统计信息的表，以便了解是否需要重组。如果需要重组，则它会在内部调度对表进行传统重组。这将要求执行应用程序功能而不对正在重组的表进行写访问。

可以使用 **auto_reorg**、**auto_tbl_maint** 和 **auto_maint** 数据库配置参数来启用或禁用自动重组功能部件。

在分区数据库环境中，确定执行自动重组和启动自动重组是在目录分区上完成的。只需要在目录分区上启用数据库配置参数。将在目标表所在的所有数据库分区上运行重组。

如果您不太确定何时以及如何重组表和索引，则可以将自动重组作为整个数据库维护方案的一部分。

可以使用“控制中心”或“运行状况中心”中的“自动维护”向导来配置用于自动重组的表。

相关概念:

- 『表重新组织』（《性能指南》）

相关任务:

- 『选择重新组织表的方法』（《性能指南》）
- 『确定何时重新组织表』（《性能指南》）
- 『启用表和索引的自动重新组织』（《性能指南》）

表的自动收集统计信息

当 SQL 编译器优化 SQL 查询方案时，有关数据库表和索引的大小的统计信息将严重影响 SQL 编译器作出决定。优化器还使用有关表和索引的特定列中的数据分发的信息，如果这些列用来选择行或者连接表。优化器使用此信息来估计每个查询的备用访问方案的成本。表或索引的统计信息过时或不完整可能会导致优化器选择效率比其他备用

方法低很多的方案，因而会降低查询执行速度。此外，决定对给定工作负载收集哪些统计信息是很复杂的事情，并且通过运行 RUNSTATS 实用程序使这些统计信息保持最新也很费时。

启用自动收集统计信息之后，自动收集统计信息在后台的工作方式是确定能够提供最佳性能改进的一组最小统计信息。通过观察和了解修改表的频率以及表统计信息的更改程度来决定如何收集或更新统计信息。自动收集统计信息算法将了解一段时间以来每个表的统计信息更改的快慢程度，并在内部安排相应地执行 RUNSTATS。

启动此功能部件不会影响正常的数据库维护活动（例如，使用 RUNSTATS 实用程序、REORG 实用程序，或者改变或废弃表）。

如果您不太确定收集数据库中的表的统计信息的频率，则可将自动收集统计信息功能部件作为整个数据库维护方案的一部分。

可以使用 **auto_runstats**、**auto_tbl_maint** 和 **auto_maint** 数据库配置参数来启用或禁用自动收集统计信息功能部件。或者，可以使用“控制中心”或“运行状况中心”中的“配置自动维护”向导来启用自动收集统计信息。

在分区数据库环境中，确定执行自动收集统计信息和启动自动收集统计信息是在目录分区上完成的。只需要在目录分区上启用 **auto_runstats** 配置参数。实际的统计信息收集是通过 RUNSTATS 完成的，并按如下所示进行收集:

1. 如果目录分区具有表数据，则在该目录分区上收集统计信息。RUNSTATS 始终会在启动了RUNSTATS 并且包含表数据的数据库分区上收集统计信息。
2. 否则，会在数据库分区列表中的第一个数据库分区上收集统计信息。

相关概念:

- 『目录统计信息』（*《性能指南》*）

相关任务:

- 『收集目录统计信息』（*《性能指南》*）

使用自动收集统计信息的自动统计信息概要分析

丢失或过期的统计信息可使优化器采用较慢的查询方案。一定要注意，并非所有统计信息对于给定的工作负载都很重要。例如，有关不出现在任何查询谓词中的列的统计信息不大可能有任何影响。有时，需要有关某几列的统计信息（列组统计信息）以便调整这些列之间的依赖性。

自动统计信息概要分析将通过只考虑先前查询中使用的那些列和了解发生评估错误的那些列或列组合来分析优化器行为。为了检测错误以及建议或更改统计性的概要文件，统计性的概要文件生成器将收集在编译查询时收集的信息以及运行查询时积累的信息。在运行了查询之后执行操作并且最终在选择并运行了方案之后此方法才起作用。

自动统计信息概要分析功能通过检测过时的、丢失的以及不正确指定的统计信息以及根据查询反馈信息来生成统计概要文件来建议如何使用 RUNSTATS 实用程序来收集统计信息。

如果能够满足您的需要，则可以将自动统计信息概要分析功能部件作为整个数据库维护方案的一部分。

自动进行统计信息概要分析会与自动收集统计信息交互，并且会建议何时收集统计信息。

可以使用 **auto_stats_prof**、**auto_tbl_maint** 和 **auto_maint** 数据库配置参数来启用或禁用自动统计信息概要分析功能部件。如果还启用了 **auto_prof_upd** 数据库配置参数，则使用生成的统计概要文件来更新 RUNSTATS 用户概要文件。对于分区数据库环境，或者当启用了对称多处理器（SMP）并行性（也称为分区内并行性）时，自动统计信息概要分析不可用。

相关概念:

- 『目录统计信息』（*《性能指南》*）

相关任务:

- 『收集目录统计信息』（*《性能指南》*）

自动收集统计信息和概要分析使用的存储器

自动进行统计信息收集和重组功能部件将工作数据存储在数据库的表中。这些表是在 SYSTOOLSPACE 表空间中创建的。当激活数据库时，会自动使用缺省选项来创建 SYSTOOLSPACE 表空间。这些表的存储器需求与数据库中表的数目成比例，应该按每个表大约 1 KB 来计算。如果这对于数据库来说太大了，则您可能想删除并亲自重新创建表空间，然后再适当地分配存储器。将自动重新创建表空间中的自动维护和运行状况监视器表。当删除表空间时，在这些表中捕获的任何历史记录都会丢失。

维护窗口

维护窗口是自动维护活动的运行时间段，由用户定义。这与任务时间表不同。在维护窗口内，不必运行每项自动维护活动。而是，DB2 根据所要运行的每项维护活动的需求来对系统进行评估。如果未符合维护需求，就运行该维护活动。如果数据库的维护状态良好，则不运行维护活动。

您可能需要确定运行自动维护活动的时间。运行自动维护活动（备份、统计信息收集、统计信息概要分析和重组）将消耗系统上的资源，并且可能会影响数据库的性能。当自动重组和脱机数据库备份运行时，这些实用程序还会限制对表和数据库的访问。因此，需要提供适当的时间段，以便可以在内部安排由 DB2 数据库管理器运行。可以使用“控制中心”或“运行状况中心”中的“自动维护”向导来将这些时间段指定为脱机和联机维护时间段。

脱机数据库备份以及表和索引重组是在脱机维护时间段内运行的。即使超过了指定的时间段，这些功能部件也将运行直到完成为止。内部调度机制经过一段时间就会了解并估计作业完成时间。如果对于特定数据库备份或重组活动来说脱机时间段太短了，则调度程序下一次将不启动作业，并依赖运行状况监视器来提供需要延长脱机维护时间段的通知。

自动进行统计信息收集和概要分析以及联机数据库备份都是在联机维护时间段内运行的。为了使它们对系统的影响最小，将由适当的实用程序调节机制来调节它们。内部调度机制使用联机维护时间段来启动联机作业。即使超过了指定的时间段，这些功能部件也将运行直到完成为止。

相关概念:

- 第 27 页的『关于自动维护功能』
- 第 33 页的『脱机维护』
- 第 33 页的『联机维护』

脱机维护

脱机维护活动是只有一定程度地中断用户对数据库的访问才能运行的维护活动。影响用户访问的程度取决于所运行的维护活动。

- 在脱机备份期间，没有任何应用程序可以与数据库连接。任何当前已连接的应用程序都将被强制断开连接。
- 在脱机数据碎片整理（表或索引重组）期间，应用程序可以访问数据库表中的数据，但无法进行更新。

注: 数据访问优化维护活动（运行统计）只能以联机方式执行。

相关概念:

- 第 27 页的『关于自动维护功能』
- 第 32 页的『维护窗口』
- 第 33 页的『联机维护』

联机维护

联机维护活动是可以在用户已连接到数据库的情况下运行的维护活动。运行联机维护活动时，允许任何当前已连接的应用程序保持连接，并允许建立新连接。

相关概念:

- 第 27 页的『关于自动维护功能』
- 第 32 页的『维护窗口』
- 第 33 页的『脱机维护』

第 3 章 并行数据库系统

本章讨论运用不同的方法分割和检索数据，以提高应用程序对数据的访问速度并减少响应的时间。其中包含数据分发、表分区、 并行性以及关于使用一个和多个处理器的信息。

并行性

任务（例如，数据库查询）的各个组件可并行运行以大幅提高性能。任务的性质、数据库配置和硬件环境都确定 DB2 数据库产品将如何以并行方式执行任务。这些注意事项是相关的，在决定数据库的物理和逻辑设计方案时，应当一起考虑它们。DB2 数据库系统支持下列类型的并行性:

- I/O
- 查询
- 实用程序

输入／输出并行性

当一个表空间有多个容器时，数据库管理器可以利用*并行 I/O*。并行 I/O 指的是同时处理对两个或多个 I/O 设备的写入或读取；这能够显著地改进吞吐量。

查询并行性

有两种类型的查询并行性：查询间并行性和查询内并行性。

*查询间并行性*是指数据库同时接受多个应用程序查询的能力。每个查询以独立于其他查询的方式运行，但 DB2 同时运行所有查询。DB2 数据库产品始终支持这种类型的并行性。

*查询内并行性*是指使用*分区内并行性*和／或*分区间并行性*来同时处理单个查询的各部分。

分区内并行性

*分区内并行性*是指将一个查询分为多个部分的能力。某些 DB2 实用程序也执行此类型的并行性。

分区内并行性将通常认为的单个数据库操作（例如，创建索引、装入数据库或 SQL 查询）细分成多个部分，其中的大部分或全部操作可以在*单个数据库分区内*以并行方式运行。

第 36 页的图 13 显示了一个查询，它被分为可并行运行的四个部分，返回结果的速度比按串行方式运行该查询的速度快得多。这几部分互为副本。要利用分区内并行性，必须适当地配置数据库。您可以选择并行度，或由系统为您选择。并行度表示一个查询中并行运行的部分数。

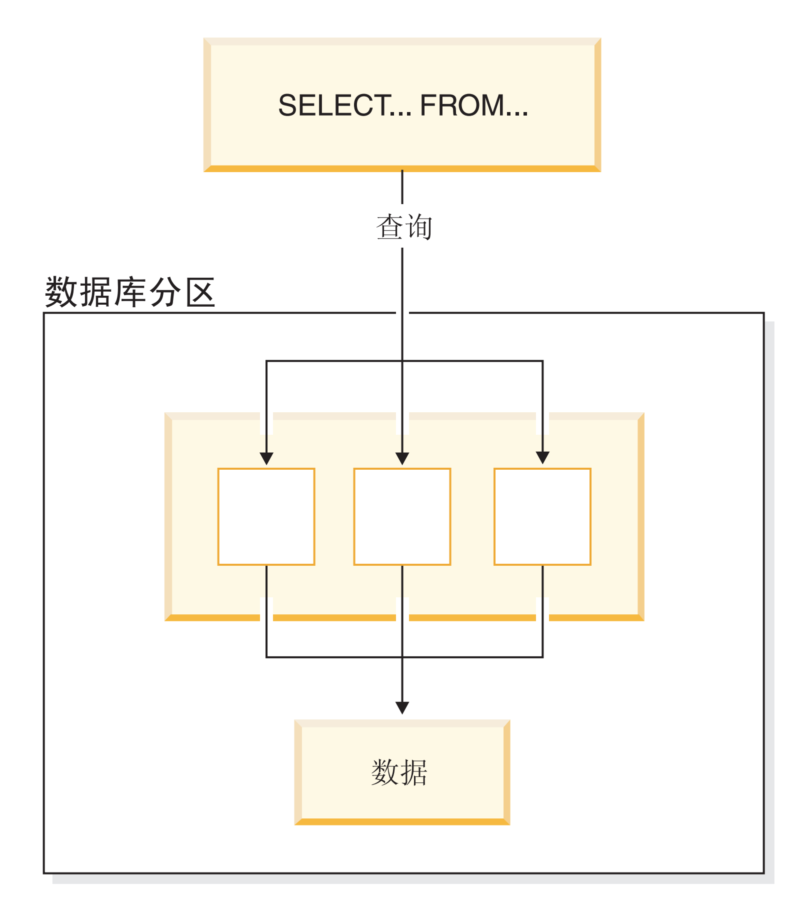


图 13. 分区内并行性

分区间并行性

*分区间并行性*是指将一个查询分为多个部分并将这几部分置于一个分区数据库的多个分区（位于一台或多台机器上）上的能力。该查询以并行方式运行。某些 DB2 实用程序也执行此类型的并行性。

分区间并行性将通常认为的单个数据库操作（例如，创建索引、装入数据库或 SQL 查询）细分成多个部分，其中的大部分或全部操作可以在*一台或多台机器上的一个分区数据库的多个分区中*以并行方式运行。

第 37 页的图 14 显示了一个查询，它被分为可并行运行的四个部分，返回结果的速度比在单个数据库分区上按串行方式运行该查询的速度快得多。

并行度在很大程度上取决于您创建的数据库分区数和您定义数据库分区组的方式。

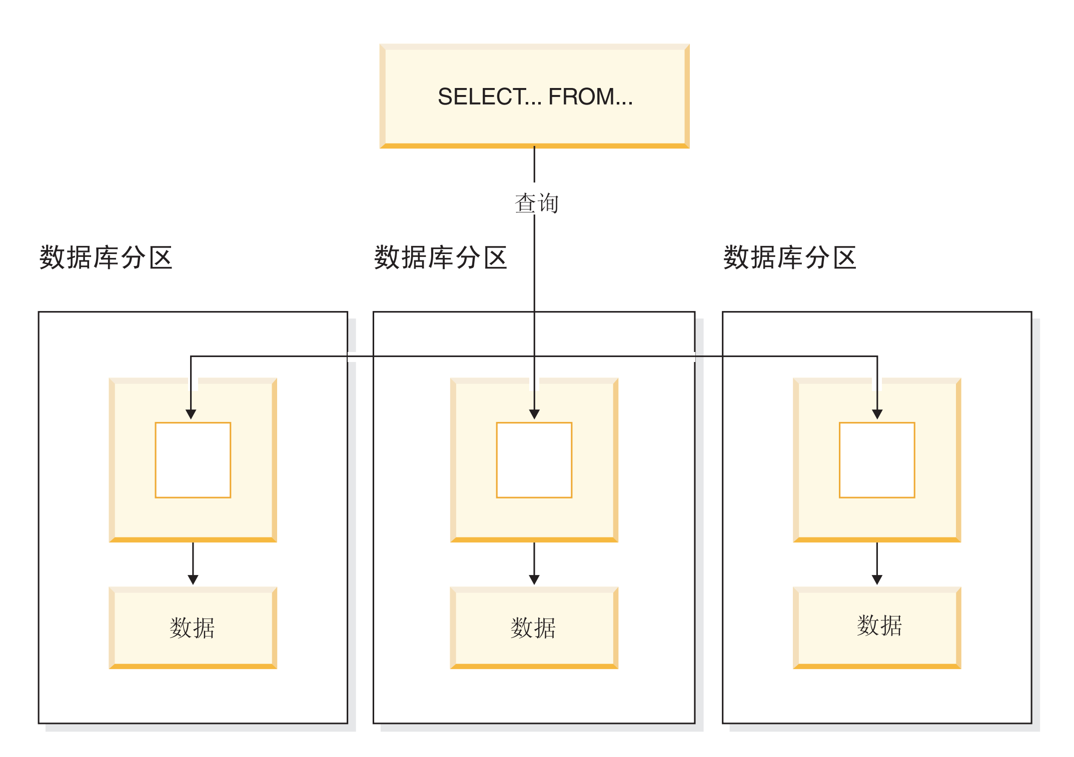


图 14. 分区间并行性

同时使用分区内和分区间并行性

可以同时使用分区内并行性和分区间并行性。此组合提供了两种并行性，这也使处理查询的速度显著加快。

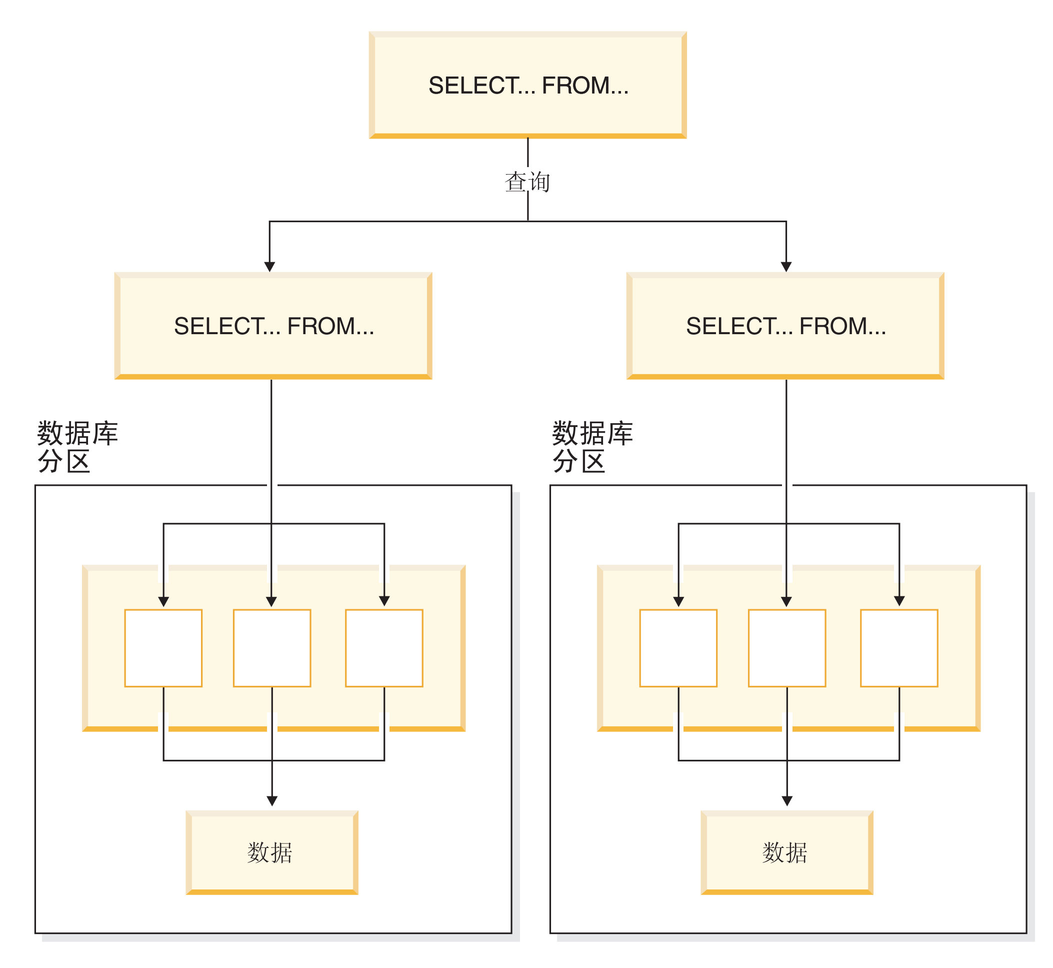


图 15. 同时使用分区间并行性和分区内并行性

实用程序并行性

DB2 实用程序可以利用分区内并行性。它们还可以利用分区间并行性；前提是存在多个数据库分区，而实用程序在每个数据库分区内并行执行。

Load 实用程序可以利用分区内并行性和 I/O 并行性。装入数据是一个大量使用 CPU 的任务。Load 实用程序利用多个处理器来执行如解析和格式化数据这类任务。它也可使用并行 I/O 服务器来以并行方式将数据写入容器中。

在分区数据库环境中，LOAD 命令通过在该表所在的每个数据库分区上进行并行调用来利用分区内、分区间和 I/O 并行性。

在创建索引期间，可并行执行数据的扫描和后续排序。在创建索引时，DB2 系统既利用 I/O 并行性又利用分区内并行性。这有助于在发出 CREATE INDEX 语句时、重新启动期间（若一个索引标记为无效）及数据重组期间加快索引创建的速度。

备份和复原数据任务是繁重地涉及 I/O 的任务。在执行备份和恢复操作时，DB2 系统既利用 I/O 并行性又利用分区内并行性。备份操作通过以并行方式读取多个表空间容器并以并行方式异步写入多备份介质，来利用 I/O 并行性。

相关概念:

• 第 40 页的『数据库分区和处理器环境』

分区数据库环境

DB2 数据库 Linux 版、UNIX 版和 Windows 版将数据库管理器扩展至并行多分区环境。*数据库分区*是数据库的一部分，它由其自己的数据、索引、配置文件和事务日志组成。数据库分区有时称为节点或数据库节点。分区数据库环境是支持将数据分布到各数据库分区上的数据库安装。

*单一分区数据库*是只有一个数据库分区的数据库。该数据库中的所有数据都存储在这个单一数据库分区中。对于这种情况，数据库分区组（如果存在的话）未提供任何其他功能。

*多分区数据库*是有两个或更多个数据库分区的数据库。表可以存储在一个或多个数据库分区中。当表存储在包含多个数据库分区的数据库分区组中时，它的某些行存储在一个数据库分区中，而其他行存储在其他数据库分区中。

通常，在每台物理机器上都存在单个数据库分区，而在每个数据库分区中，数据库管理器使用每个系统上的处理器来管理该数据库的全部数据中属于该分区的那一部分。

由于数据分布在各数据库分区中，所以可以使用多台物理机器上多个处理器的能力来满足对信息的请求。数据检索和更新请求被自动分解成子请求，并在适用的数据库分区中并行执行。数据库分布在各数据库分区中的这个事实对于发出 SQL 语句的用户是透明的。

用户交互通过一个数据库分区发生，该数据库分区称为该用户的*协调程序分区*。协调程序分区与应用程序在同一个数据库分区中运行，对于远程应用程序来说，协调程序分区在该应用程序所连接至的数据库分区中运行。任何数据库分区都可用作协调程序分区。

DB2 允许将数据存储在数据库的多个数据库分区中。这意味着该数据以物理方式存储在多个数据库分区中，但是对其进行访问时，仍可以将数据视为位于同一位置。访问多分区数据库中的数据的应用程序和用户不需要知道该数据的物理位置。

该数据在物理上是分离的，但却将它作为一个逻辑整体来使用和管理。用户可以通过声明分布键来选择如何分布数据。用户还可以通过选择表空间和应存储数据的相关数据库分区组，确定数据可以分布在哪些数据库分区上以及分布在多少个数据库分区上。可以使用 DB2 设计顾问程序来完成有关分布和复制的建议。另外，将可更新的分布图与散列算法一起使用，来指定分布键值到数据库分区的映射，以确定每行数据的位置和检索。因此，可以将大型表的工作负载分布在多分区数据库上，并将较小的表存储在一个或多个数据库分区中。每个数据库分区都有它所存储数据的本地索引，这可以提高本地数据的访问性能。

并不是必须将所有表都分布到数据库的所有数据库分区中。DB2 支持*部分分区*，这意味着可以将表及其表空间分布到系统中的数据库分区的子集上。

当您想要将表放在每个数据库分区中时，可考虑的替代方法是使用具体化查询表，然后复制那些表。您可以创建包含所需信息的具体化查询表，然后将它复制到每个数据库分区。

数据库分区和处理器环境

本节对下列硬件环境提供了一个概述:

- 单处理器上的单个数据库分区（单处理器）
- 具有多处理器的单个数据库分区（SMP）
- 多个数据库分区配置
  - 具有单处理器的数据库分区（MPP）
  - 具有多处理器的数据库分区（SMP 集群）
  - 逻辑数据库分区

下面讨论每种环境的容量和可伸缩性。*容量*是指能访问数据库的用户数和应用程序数。这很大程度上取决于内存、代理程序数、锁定数、I/O 和存储器管理。*可伸缩性*是指一个数据库随着其增长而继续显示出相同的操作特征和响应时间的能力。

单处理器上的单个数据库分区

此环境由内存和磁盘组成，但仅包含一个 CPU（请参阅图 16）。可使用许多不同的名称来称呼它，包括独立数据库、客户机／服务器数据库、串行数据库、单处理器系统和单节点或非并行环境。

此环境中的数据库可满足一个部门或小办公室的需要，其中的数据和系统资源（包括一个单处理器或 CPU）由单数据库管理器来管理。

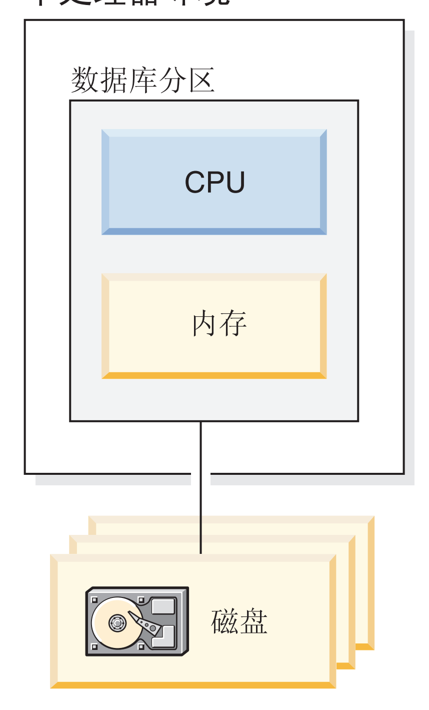


图 16. 单处理器上的单个数据库分区

容量和可伸缩性

在此环境中可以添加更多的磁盘。如果让每个磁盘拥有一个或多个 I/O 服务器，则多个 I/O 操作可同时执行。

单处理器系统受处理器可处理的磁盘空间容量的限制。无论可以添加的其他组件（如内存或磁盘）如何，随着工作负载的增加，单个 CPU 可能无法更快地处理用户请求。若已达到最大容量或最大可伸缩性，则可考虑移到一个有多处理器的单个数据库分区系统中。

具有多处理器的单个数据库分区

此环境通常由同一台机器中几个能力相等的处理器组成（请参阅图 17），称为*对称多处理器（SMP）*系统。像磁盘空间和内存这类资源是*共享的*。

利用可用的多个处理器，可以更快地完成不同的数据库操作。DB2 数据库系统还可把单个查询的工作分布在可用的处理器中，以提高处理速度。其他数据库操作，例如，装入数据、备份和复原表空间以及对现有数据创建索引，都可以利用多个处理器。

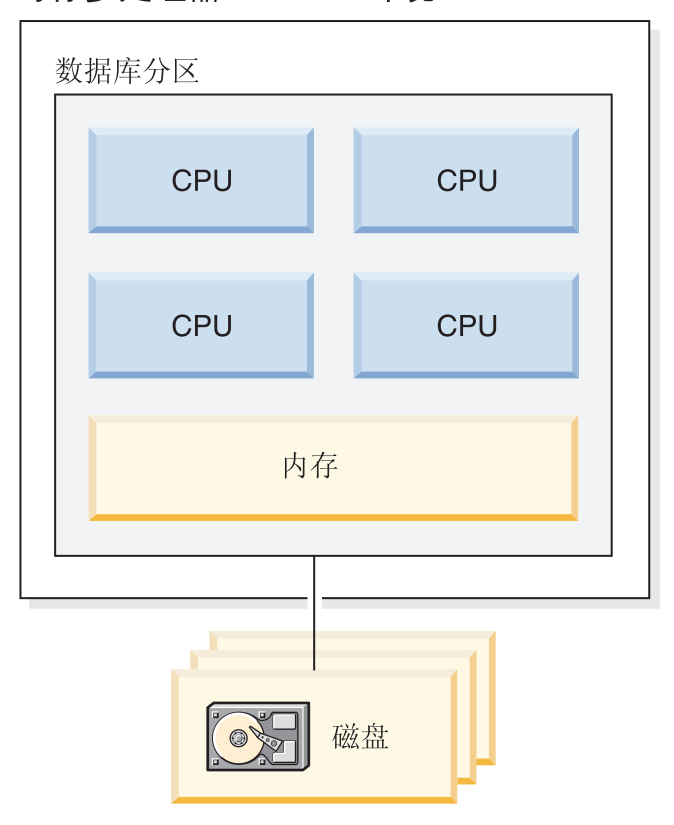


图 17. 单一分区数据库对称多处理器环境

容量和可伸缩性

在此环境中可以添加更多的处理器。然而，由于不同的处理器可能会试图访问同一数据，所以随着您公司业务量的增加，此环境也可能遇到限制。利用共享内存和共享磁盘，可有效地共享所有数据库数据。

可以通过增加磁盘数来增加与处理器相关的数据库分区的 I/O 容量。可以建立 I/O 服务器以专门处理 I/O 请求。如果让每个磁盘拥有一个或多个 I/O 服务器，则多个 I/O 操作可同时执行。

若已达到最大容量或最大可伸缩性，则可考虑移到有多个数据库分区的系统中。

多个数据库分区配置

可以将一个数据库分布在多个数据库分区中，每个数据库分区在它自己的机器上。可将有多个数据库分区的多台机器编组在一起。本节描述下列数据库分区配置:

- 具有单处理器的系统上的数据库分区
- 具有多处理器的系统上的数据库分区
- 逻辑数据库分区

具有单处理器的数据库分区

在此环境中，有许多数据库分区。每个数据库分区都位于它自己的机器上，而且它有自己的处理器、内存和磁盘（图 18）。所有机器通过通信工具连接在一起。可使用许多不同的名称来称呼此环境，包括：集群、单处理器集群、大规模并行处理（MPP）环境和“不共享”配置。最后一个名称准确地反映了此环境中的资源安排。与 SMP 环境不同，MPP 环境没有共享的内存或磁盘。MPP 环境消除了由共享内存和磁盘带来的限制。

一个分区数据库环境允许一个数据库保持逻辑上的整体性，但它实际上分布于多个数据库分区中。数据是分布的这一事实对大多数用户是透明的。可以在数据库管理器之间划分工作；每个数据库分区中的每个数据库管理器都只为该数据库中它自己的那部分工作。

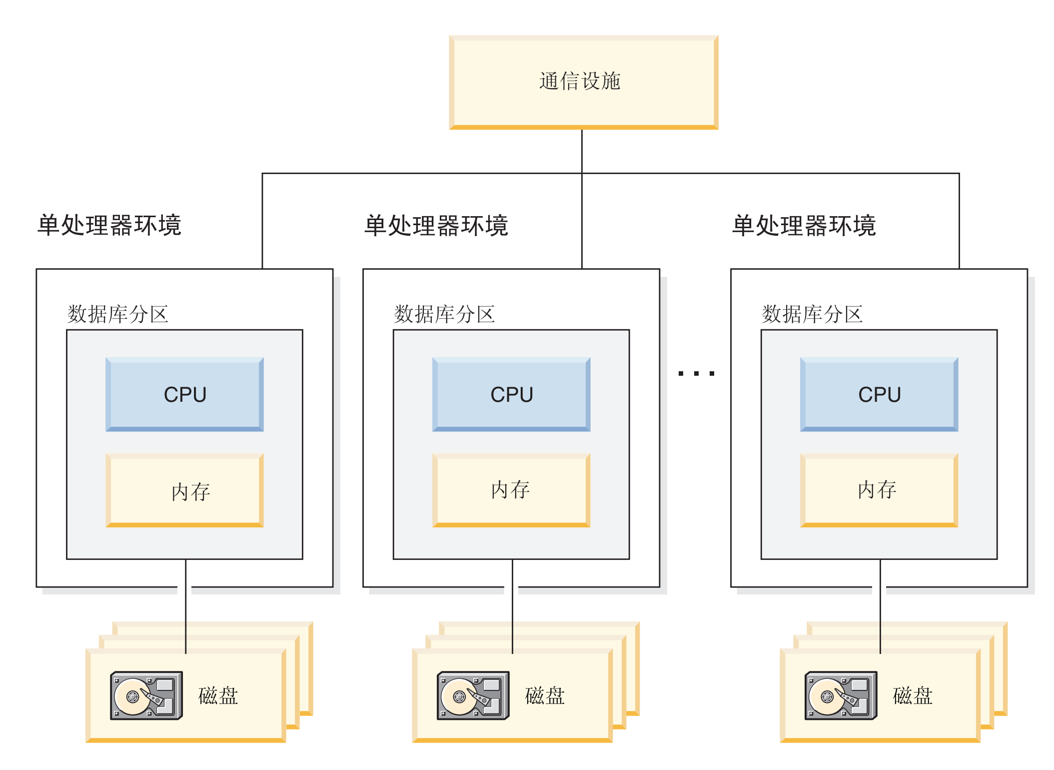


图 18. 大规模并行处理（MPP）环境

容量和可伸缩性: 在此环境中可以向您的配置添加更多的数据库分区（节点）。在某些平台上（例如，RS/6000® SP™），最大数目是 512 个节点。然而，在管理很多机器和实例时，可能存在实际的限制。

若已达到最大容量或最大可伸缩性，则可考虑移到每个数据库分区都有多个处理器的系统中。

具有多处理器的数据库分区

每个数据库分区具有单处理器的替代配置是，每个数据库分区具有多处理器的配置。这称为 *SMP 集群*（图 19）。

此配置结合了 SMP 和 MPP 并行性的优点。这表示一个查询可以在跨多处理器的单个数据库分区中执行。亦即一个查询可以用并行方式在多个数据库分区中执行。

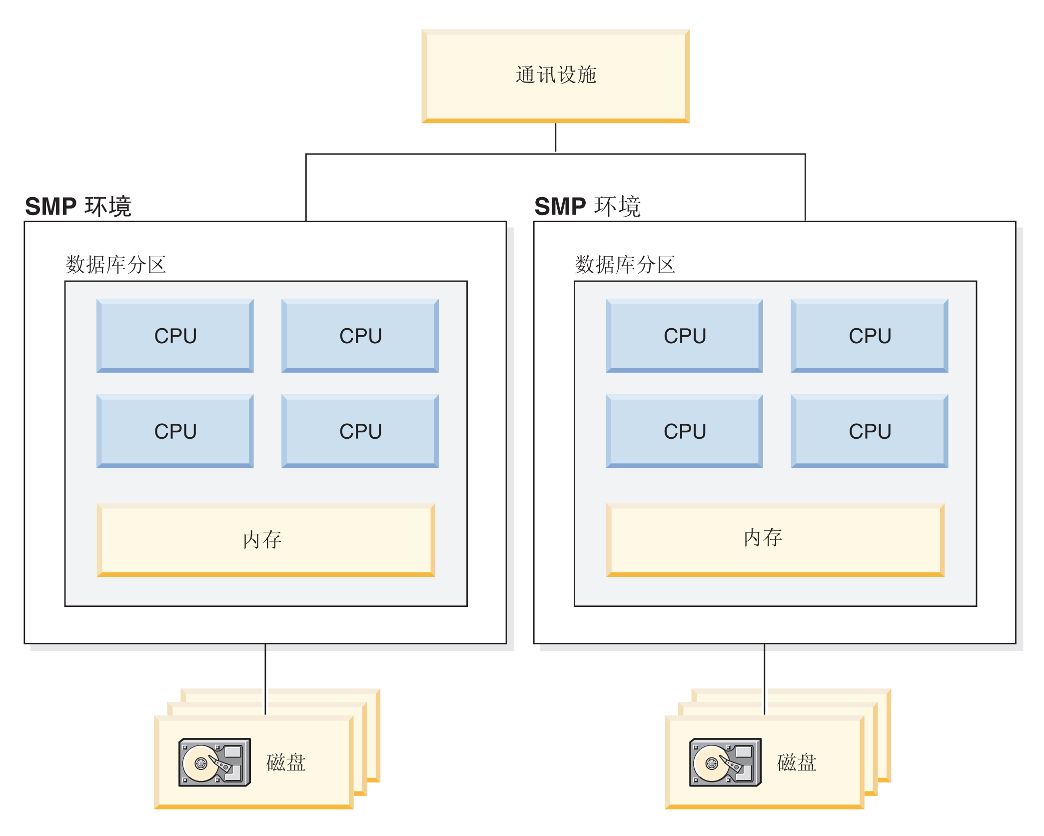


图 19. 集群中的几个对称多处理器（SMP）环境

容量和可伸缩性: 在此环境中，可以添加更多的数据库分区，并可以向现有数据库分区添加更多的处理器。

逻辑数据库分区

逻辑数据库分区与物理分区的不同之处在于逻辑分区未被授予对整台机器的控制权。虽然机器已共享资源，但是数据库分区不共享资源。处理器是共享的，但磁盘和内存却不共享。

逻辑数据库分区提供了可伸缩性。在多个逻辑分区上运行的多个数据库管理器可以比一个单数据库管理器更能充分利用可用的资源。第 44 页的图 20 举例说明通过添加更多的数据库分区可以在一台 SMP 机器上获得更大的可伸缩性；特别是对那些具有许多处

理器的机器而言更是如此。通过分配数据库，可以分别对每个数据库分区进行管理和恢复。

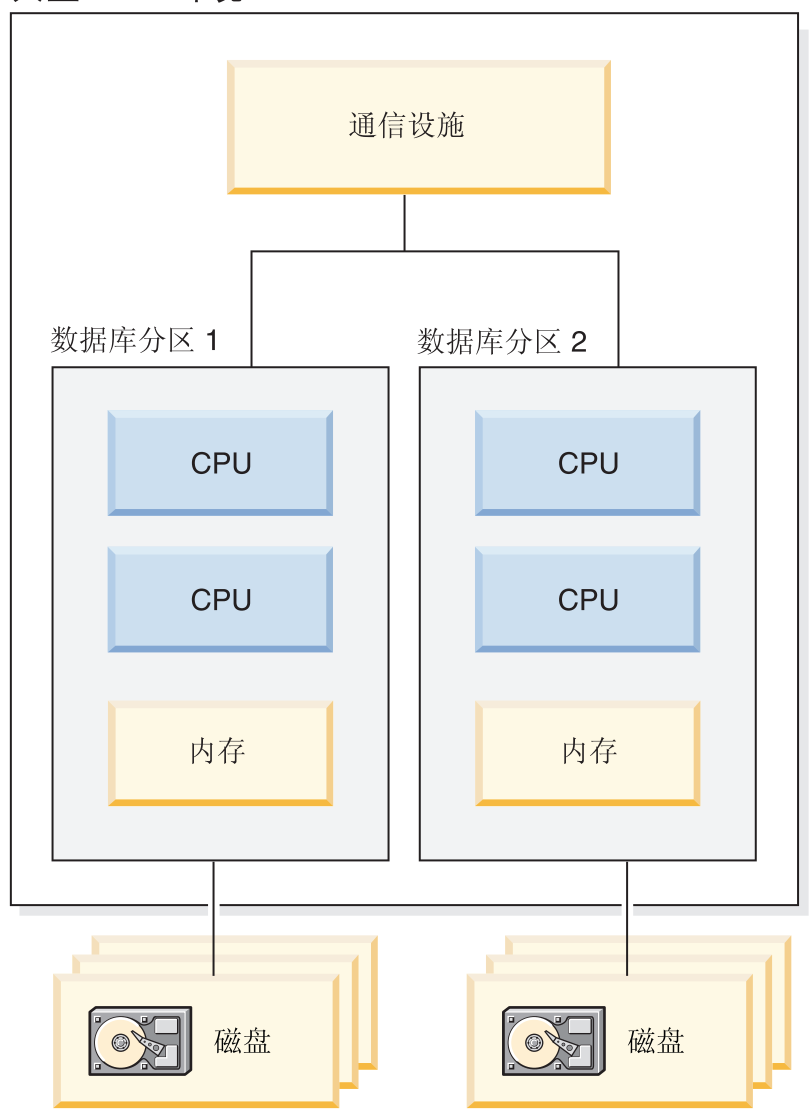


图 20. 带有对称多处理器环境的分区数据库

第 45 页的图 21 举例说明可以扩大图 20 中显示的配置以增强处理能力。

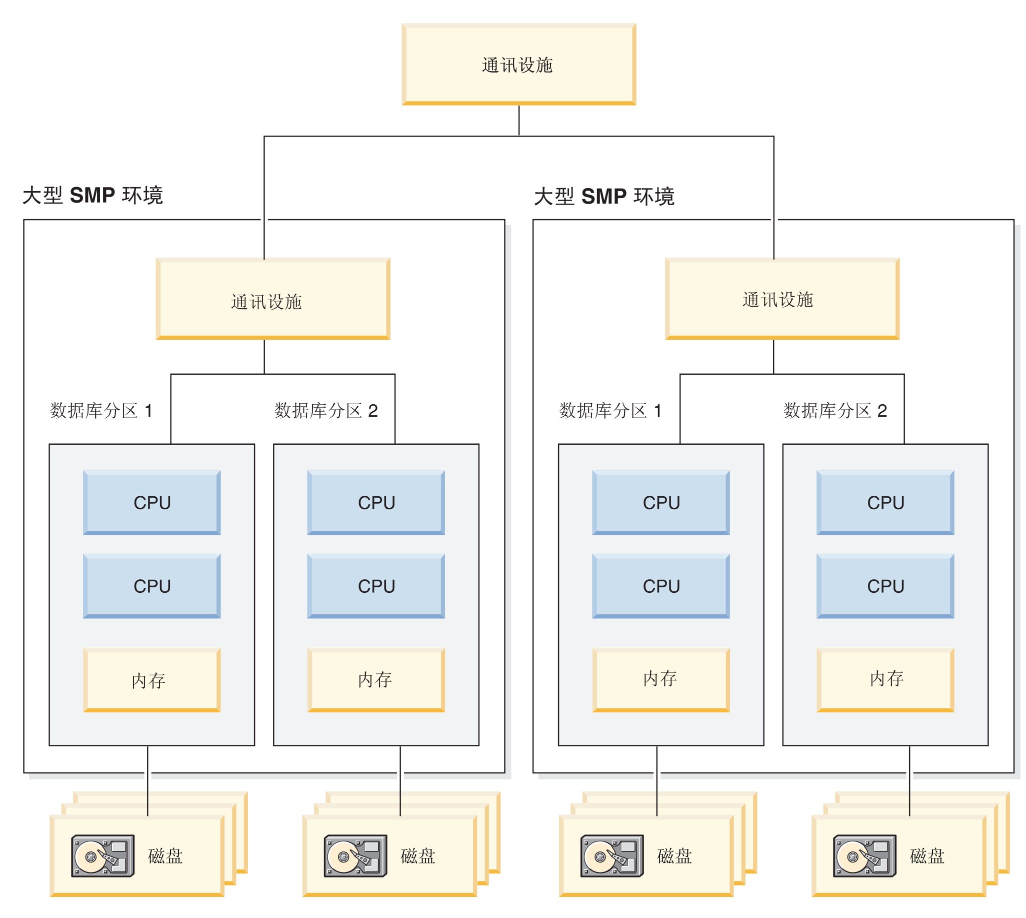


图 21. 带有集中在一起的对称多处理器环境的分区数据库

注： 两个以上的数据库分区同时存在于同一台机器上的能力（不考虑处理器的数量），使得可以更灵活地设计高可用性配置和故障转移策略。机器发生故障之后，可以将一个数据库分区自动移至已包含同一数据库的另一个数据库分区的另一台机器上，然后重新启动该分区。

最适合每个硬件环境的并行性总结

下表总结了最适合于利用各种硬件环境的并行性的类型。

表 3. 每种硬件环境中的可能并行性类型

硬件环境	I/O 并行性	查询内并行性	
		分区内并行性	分区间并行性
单个数据库分区，单处理器	是	否（1）	否
单个数据库分区，多处理器（SMP）	是	是	否
多个数据库分区，单处理器（MPP）	是	否（1）	是
多个数据库分区，多处理器（SMP 集群）	是	是	是
逻辑数据库分区	是	是	是
注： (1) 甚至是在单处理器系统中，使用一个配置参数将并行度设置为大于 1 的值也可能获得好处，特别是在执行的查询并未充分利用 CPU 时（例如，若它们受 I/O 约束）。			

相关概念:

- 第 35 页的『并行性』

第 2 部分 数据库设计

第 4 章 逻辑数据库设计

设计数据库时，您希望精确地表示您的环境以作为将来扩展的基础。另外，您的数据库设计应该维持数据的一致性和完整性。要达到此目的，可以设计一个减少冗余并消除在更新数据库期间可能发生的异常的数据库。本章的主题将讨论逻辑数据库设计的元素。

要在数据库中记录的内容

数据库设计的第一步是标识要存储在数据库表中的数据的类型。一个数据库包括一个组织或企业中的*实体*以及它们之间的关系的信息。在关系数据库中，将*实体*表示为*表*。

一个*实体*是您想要存储其信息的一个人、一个对象或一个概念。样本表中描述的某些实体是职员、部门和项目。

在 Employee 样本表中，实体“职员”具有*属性*或特性，例如，职员号、姓名、工作部门和工作描述。这些特性显示为 EMPNO、FIRSTNME、LASTNAME、WORKDEPT 和 JOB *列*。

实体“职员”的*具体值*由一个职员的所有列中的值组成。每个职员有一个唯一的职员号（EMPNO），该号码可用来标识实体“职员”的具体值。一个表中的每一行表示一个实体或关系的一个具体值。例如，在下表中，第一行中的值描述名为 Haas 的职员。

表 4. 职员实体及其属性的具体值

EMPNO	FIRSTNME	LASTNAME	WORKDEPT	JOB
000010	Christine	Haas	A00	President
000020	Michael	Thompson	B01	Manager
000120	Sean	O’Connell	A00	Clerk
000130	Dolores	Quintana	C01	Analyst
000030	Sally	Kwan	C01	Manager
000140	Heather	Nicholls	C01	Analyst
000170	Masatoshi	Yoshimura	D11	Designer

随着支持非传统数据库应用程序（例如，多媒体）的需要日益增长。可能要考虑一些属性以便支持多媒体对象，例如，文档、视频或混合媒体、图像和语音。

在一个表中，某一行的每一列都以某种方式与该行的所有其他列相关。样本表中表达的一些关系如下:

- 职员被分配至部门
  - Dolores Quintana 被分配至部门 C01
- 职员从事一个工作
  - Dolores 从事分析员的工作
- 职员管理部门

– Sally 管理部门 C01

“职员”和“部门”是实体; Sally Kwan 是“职员”的一个具体值的一部分，而 C01 是“部门”的一个具体值的一部分。相同的关系应用于一个表的每一行中的相同列。例如，一个表的某行表示的关系是 Sally Kwan 管理部门 C01; 而另一行表示的关系是 Sean O’Connell 是部门 A00 中的职员。

一个表中包含的信息取决于要表示的关系、需要的灵活度和期望的数据检索速度。

除了标识企业中的实体关系外，还需标识其他类型的信息，如应用于该数据的业务规则。

相关概念:

- 第 52 页的『列定义』
- 第 50 页的『数据库关系』

数据库关系

可以在数据库中定义几种类型的关系。考虑职员和部门之间的可能关系。

一对多和多对一关系

一个职员只能在一个部门工作; 对于职员，此关系是*单值的*。另一方面，一个部门可有许多职员; 对于部门，此关系是*多值的*。职员（单值的）和部门（多值的）之间的关系是*一对多*的关系。

要为每个一对多和每个多对一关系定义表:

1. 将关系的“多”方是相同实体的所有关系分组。
2. 为该组中的所有关系定义单个表。

在以下示例中，第一个和第二个关系的“多”方是“职员”，因此我们定义了一个职员表 EMPLOYEE。

表 5. 多对一关系

实体	关系	实体
职员	被分配到	部门
职员	从事	工作
部门	报告给	（管理）部门

在第三个关系中，“部门”是“多”方，因此我们定义了一个部门表 DEPARTMENT。

下列各表说明了这些不同的关系。

EMPLOYEE 表:

EMPNO	WORKDEPT	JOB
000010	A00	President
000020	B01	Manager
000120	A00	Clerk
000130	C01	Analyst
000030	C01	Manager
000140	C01	Analyst
000170	D11	Designer

DEPARTMENT 表:

DEPTNO	ADMRDEPT
C01	A00
D01	A00
D11	D01

多对多关系

两个方向都是多值的关系是多对多关系。一个职员可以处理多个项目，而一个项目可以有多个职员。问题“Dolores Quintana 在处理什么？”和“谁在处理项目 IF1000？”都得到多个答案。在一个表中，对每个实体（“职员”和“项目”）使用一列，可以表示多对多关系，如下面示例所示。

下表显示如何在一个表中表示多对多关系（一个职员可以参与多个项目，而一个项目可以有多个职员在参与）。

职员活动（EMP_ACT）表:

EMPNO	PROJNO
000030	IF1000
000030	IF2000
000130	IF1000
000140	IF2000
000250	AD3112

一对一关系

一对一关系在两个方向都是单值的。一个经理管理一个部门；一个部门只有一个经理。问题“谁是部门 C01 的经理？”和“Sally Kwan 管理哪个部门？”都有单个答案。可将该关系指定给 DEPARTMENT 表或 EMPLOYEE 表。因为所有部门都有经理，但不是所有职员都是经理，因此将经理添加至 DEPARTMENT 表是最合乎逻辑的，如以下示例中所示。

下表显示了一对一关系的表示法。

DEPARTMENT 表:

DEPTNO	MGRNO
A00	000010
B01	000020
D11	000060

确保相等值表示同一实体

您可以有多个表，它们描述同一组实体的属性。例如，EMPLOYEE 表显示职员被分配至的部门的编号，DEPARTMENT 表显示分配到每个部门的经理。要同时检索这两组属性，可用匹配的列将这两个表连接在一起，如以下示例中所示。WORKDEPT 和 DEPTNO 中的值表示相同的实体，并表示 DEPARTMENT 和 EMPLOYEE 表之间的*连接路径*。

DEPARTMENT 表:

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
D21	Administration Support	000070	D01

EMPLOYEE 表:

EMPNO	FIRSTNAME	LASTNAME	WORKDEPT	JOB
000250	Daniel	Smith	D21	Clerk

当从多个表检索一个实体的信息时，确保相等的值表示相同的实体。连接列可有不同的名称（如上一个示例中的 WORKDEPT 和 DEPTNO），也可以有相同的名称（如 Department 表和 Project 表中的 DEPTNO 列）。

相关概念:

- 第 52 页的『列定义』
- 第 49 页的『要在数据库中记录的内容』

列定义

在关系表中，表中的每个数据行都是相关数据值的集合。每一行中的每个数据都有一些特征。列用来标识每个数据并对它们进行分类。

一个表的每一列都必须具有对于该表唯一的名称。

*数据类型*和*长度*指定对该列有效的数据类型和最大长度。可以从数据库管理器提供的那些类型中选择数据类型；也可以选择创建自己的用户定义的类型。

数据类型的示例有：数字、字符串、双字节（或图形）字符串、日期时间和二进制字符串。

大对象（LOB）数据类型支持多媒体对象，如文档、视频、图像和语音。这些对象是使用下列数据类型实现的:

- *二进制大对象*（BLOB）字符串。BLOB 的示例是职员的相片、语音和视频。
- *字符大对象*（CLOB）字符串，它的字符序列可以是单字节字符或多字节字符，或这两者的组合。CLOB 的一个示例是职员的简历。
- *双字节字符大对象*（DBCLOB）字符串，它的字符序列是双字节字符。DBCLOB 的一个示例是日语简历。

用户定义的类型（UDT）是从现有类型派生的一种数据类型。您可能需要定义从现有数据类型派生并与现有数据类型共享特征，但仍被视为与它们不同且不兼容的类型。

*结构化类型*是一种用户定义的类型，其结构在数据库中定义。它包含一系列命名的*属性*，每个属性都有一个数据类型。可将结构化类型定义为称为*超类型*的另一种结构化类型的*子类型*。子类型继承它的超类型的所有属性，并可定义附加属性。与一个公共超类型相关的结构化类型集合称为*类型层次结构*，没有任何超类型的超类型称为该类型层次结构的*根类型*。

可以将结构化类型用作表或视图的类型。结构化类型的属性的名称和数据类型，与对象标识一起，成为这个*类型表*或*带类型视图*的列的名称和数据类型。可以将类型表或带类型视图的行当作结构化类型实例的表示法。

不能将结构化类型用作表或视图的列的数据类型。也不支持将整个结构化类型的实例作为应用程序中的一个主变量来检索。

*引用类型*是结构化类型的辅助类型。类似于单值类型，引用类型是一个标量类型，它与一种内置数据类型共享一个公共的表示法。在类型层次结构中的所有类型都共享这同一个表示法。引用类型的表示法是在创建类型层次结构的根类型时定义的。当使用引用类型时，将结构化类型指定为该类型的参数。此参数称为该引用的*目标类型*。

引用的目标始终是类型表或视图中的行。当使用引用类型时，可能需要定义*作用域*。作用域标识一个表（称为*目标表*）或视图（称为*目标视图*），它包含引用值的目标行。目标表或视图的类型必须与引用类型的目标类型相同。定义了作用域的引用类型的一个实例会唯一地标识类型表或带类型视图中的一行，该行称为它的*目标行*。

由于许多原因，可能会使用*用户定义的函数*（UDF），包括调用那些允许在用户定义的类型之间比较或转换的例程。UDF 扩展并增强了内置 SQL 函数提供的支持，并可在可使用内置函数的任何地方使用。有两种类型的 UDF:
- 外部函数，它是用一种编程语言编写的
- 有源函数，它将用于调用其他 UDF

例如，两个数字数据类型是“欧洲鞋码”和“美国鞋码”。这两种类型表示的都是鞋码，但是它们不兼容，因为度量基础不同所以不能进行比较。可调用用户定义的函数将一种鞋码转换为另一种。

某些列可能不是在所有行都具有有意义的值，因为:
- 列值不适用于该行。

  例如，包含职员中间姓名首字母的一列不适用于没有中间姓名首字母的职员。
- 值适用，但尚不知道该值。

  例如，MGRNO 列可能包含无效的经理号，因为该部门的上一个经理已调离，新经理尚未任命。

在这两种情况下，可以选择允许 NULL 值（指示列值是未知的或不适用的特殊值），或允许数据库管理器或应用程序指定的非 NULL 缺省值。

主键

*键*是一组可用来标识或访问特定行的列。在表、索引或引用约束的描述中标识键。相同列可以是多个键的部分。

*唯一键*是约束为其任何两个值都不相等的键。唯一键的列不能包含空值。例如，可将职员号码列定义为唯一键，因为该列中的每个值只标识唯一一个职员。任何两个职员不能有相同职员号码。

用来强制键唯一性的机制称为*唯一索引*。一个表的唯一索引是一列或若干列的有序集合，而每个值标识（在功能上确定）这些列的唯一行。唯一索引可以包含空值。

*主键*是在一个表上定义的唯一键中的一个，而且该键被选为最重要的键。一个表上只能有一个主键。

会自动为主键创建*主索引*。数据库管理器使用主索引来有效地访问表行，且主索引允许数据库管理器强制主键的唯一性。（也可以在非主键列上定义索引，以便在处理查询时高效率地访问数据。）

若一个表没有“自然”的唯一键，或者到达顺序是用于区分唯一行的方法，则使用时间戳记作为键的一部分可能有帮助。

某些样本表的主键是:

表	键列
Employee 表	EMPNO
Department 表	DEPTNO
Project 表	PROJNO

以下示例显示 PROJECT 表的一部分，包括其主键列。

表 6. PROJECT 表上的主键

PROJNO（主键）	PROJNAME	DEPTNO
MA2100	Weld Line Automation	D01
MA2110	Weld Line Programming	D11

若一个表中的每一列都包含重复的值，则不能只用一列来定义主键。具有多列的键是*组合键*。列值的组合应定义一个唯一实体。若定义组合键不太容易，可以考虑创建具有唯一值的新列。

以下示例显示包含多列的一个主键（组合键）：

表 7. EMP_ACT 表上的组合主键

EMPNO（主键）	PROJNO（主键）	ACTNO （主键）	EMPTIME	EMSTDATE（主键）
000250	AD3112	60	1.0	1982-01-01
000250	AD3112	60	.5	1982-02-01
000250	AD3112	70	.5	1982-02-01

标识候选键列

要标识候选键，选择定义唯一实体的最少数目的列。可能有多个候选键。在表 8 中，有多个候选键。EMPNO、PHONENO 和 LASTNAME 这三列都唯一地标识该职员。

表 8. EMPLOYEE 表

EMPNO（主键）	FIRSTNAME	LASTNAME	WORKDEPT（外键）	PHONENO
000010	Christine	Haas	A00	3978
000030	Sally	Kwan	C01	4738
000060	Irving	Stern	D11	6423
000120	Sean	O’Connell	A00	2167
000140	Heather	Nicholls	C01	1793
000170	Masatoshi	Yoshimura	D11	2890

从一组候选键中选择一个主键的标准应是持久性、唯一性和稳定性:

- 持久性表示每一行都存在主键值。
- 唯一性表示每一行的键值与所有其他行不同。
- 稳定性表示主键值始终不变。

在该示例的三个候选键中，只有 EMPNO 满足所有标准。一个职员在进入一家公司时，可能没有电话号码。姓可能改变，尽管它们在某个时候可能是唯一的，但不能保证始终如此。职员号码列是主键的最佳选择。一个职员只被分配一次一个唯一的号码，而且只要该职员在该公司内任职，通常不会更新该号码。因为每个职员必须有一个号码，所以职员号码列中的值是持久的。

相关概念:

- 第 55 页的『标识列』

标识列

*标识列*为 DB2 数据库 Linux 版、UNIX 版和 Windows 版提供了一种方法，可自动为表中的每一行生成唯一数值。一个表只能有一个定义了标识属性的列。标识列的示例包括订单号、职员编号、股票代码或者事故编号。

标识列的值可以是“始终生成”或“在缺省情况下生成”。

- 定义为*始终生成*的标识列不允许在 SQL 语句中覆盖值。它的值总是由 DB2 数据库管理器生成；不允许应用程序提供显式的值。不保证"始终生成"列中的值是唯一的。要保证该列中的值是唯一的，应该对该列定义唯一索引。
- 定义为*在缺省情况下生成*的标识列向应用程序提供了显式地为标识列提供一个值的方法。若未给出任何值，则 DB2 将生成一个值，但在此情况下，不能保证该值的唯一性。"在缺省情况下生成"用于数据传播（复制现有表的内容），或用于一个表的卸装和重新装入。

标识列特别适合用于生成唯一主键值。应用程序可使用标识列来避免当一个应用程序在数据库外部生成它自己的唯一计数器时可能会导致的并发性和性能问题。例如，一种常见的应用程序级实施是维护一个只有一行的表，它包含一个计数器。每个事务都锁定此表，增大该数字，然后落实；即，每次只有一个事务可以增大计数器。相反，若通过标识列维护该计数器，因为事务不锁定该计数器，所以可以获得较高级别的并发性。一个未落实的已增大该计数器的事务不会阻止后续事务也增大该计数器。

标识列的计数器的增大（或减小）独立于事务。若给定的事务两次增大一个标识计数器，则该事务可能会在生成的两个数字之间看到一个间隔，因为可能有其他事务正在增大同一标识计数器（即，将行插入同一个表中）。若一个应用程序必须要有连续范围的数字，则该应用程序应对带有标识列的表进行互斥锁定。因为会造成丢失并发性，所以必须对此决定作权衡。此外，有可能因为生成标识列的值的事务已回滚，或因为高速缓存值序列的数据库在指定所有高速缓存的值之前被取消激活，从而导致给定的标识列出现数字之间生成间隔的情况。

标识列生成的顺序数字具有下列附加属性:

- 值可以是任何小数位为零的精确数字数据类型；即，小数位为零的 SMALLINT、INTEGER、BIGINT 或 DECIMAL。（单精度和双精度浮点类型被认为是近似数字数据类型。）
- 连续值之间可以有任何指定的整数增量。缺省增量是 1。
- 标识列的计数器值是可恢复的。若发生故障，则从日志重新构造计数器值，因此可以保证继续生成唯一的值。
- 可以将标识列值存入高速缓存，以获得更好的性能。

相关概念:

- 第 54 页的『主键』

规范化

*规范化*帮助消除表数据中的冗余和不一致。它是将表精简为一组列的过程，在这组列中，所有非键列都依赖于主键列。若不是这样，则在更新期间该数据可能变得不一致。

本节简要回顾第一范式、第二范式、第三范式和第四范式的规则。表的第五范式在有关数据库设计的许多书籍中都有说明，在此就不再赘述。

格式　描述

第一范式
: 表中的每一行的每一列位置处只存在一个值，永远不会是一组值。

第二范式
: 不是键的一部分的每一列都依赖于该键。

第三范式
: 每个非键列都独立于其他非键列，并依赖于该键。

第四范式
: 没有一行包含有关实体的两个或更多个独立多值事实。

第一范式

若每个单元中都只有一个值，永远不会有一组值，则该表使用的是*第一范式*。使用第一范式的表不必满足更高的范式的标准。

例如，下表违反了第一范式，因为对于 PART 的每个具体值，WAREHOUSE 列都包含了几个值。

表 9. 违反第一范式的表

PART（主键）	WAREHOUSE
P0010	Warehouse A、Warehouse B 和 Warehouse C
P0020	Warehouse B 和 Warehouse D

以下示例显示使用第一范式的同一个表。

表 10. 符合第一范式的表

PART（主键）	WAREHOUSE（主键）	QUANTITY
P0010	Warehouse A	400
P0010	Warehouse B	543
P0010	Warehouse C	329
P0020	Warehouse B	200
P0020	Warehouse D	278

第二范式

若每个不是键一部分的列都依赖于整个键，则该表使用的是*第二范式*。

当一个非键列依赖于组合键的一部分时，违反了第二范式，如以下示例所示:

表 11. 违反第二范式的表

PART（主键）	WAREHOUSE（主键）	QUANTITY	WAREHOUSE_ADDRESS
P0010	Warehouse A	400	1608 New Field Road
P0010	Warehouse B	543	4141 Greenway Drive
P0010	Warehouse C	329	171 Pine Lane
P0020	Warehouse B	200	4141 Greenway Drive
P0020	Warehouse D	278	800 Massey Street

主键是一个组合键，它由 PART 和 WAREHOUSE 列组成。因为 WAREHOUSE_ADDRESS 列只依赖于 WAREHOUSE 的值，因此该表违反了第二范式的规则。

此设计存在下列问题:

- 对于该仓库中存储的一个部件，在每个记录中重复了该仓库地址。
- 若一个仓库的地址变更，则必须更新引用存储在该仓库中的一个部件的每一行。
- 由于存在这种冗余，该数据可能变得不一致，表现为不同的记录对相同的仓库显示不同的地址。
- 若某个时候一个仓库中没有存储部件，就可能没有哪一行记录了该仓库地址。

解决方案是将该表分割成下面两个表:

表 12. 符合第二范式的 PART_STOCK 表

PART（主键）	WAREHOUSE（主键）	QUANTITY
P0010	Warehouse A	400
P0010	Warehouse B	543
P0010	Warehouse C	329
P0020	Warehouse B	200
P0020	Warehouse D	278

表 13. 符合第二范式的 WAREHOUSE 表

WAREHOUSE（主键）	WAREHOUSE_ADDRESS
Warehouse A	1608 New Field Road
Warehouse B	4141 Greenway Drive
Warehouse C	171 Pine Lane
Warehouse D	800 Massey Street

若有两个表使用第二范式，就需要考虑性能。根据部件位置生成报告的应用程序必须将这两个表连接起来以检索相关信息。

第三范式

若每个非键列都独立于其他非键列，且只依赖于键，则该表使用的是*第三范式*。

以下示例中的第一个表包含 EMPNO 列和 WORKDEPT 列。假定添加了 DEPTNAME 列（请参阅第 59 页的表 15）。这个新列依赖于 WORKDEPT，但主键是 EMPNO。该表现在违反了第三范式。更改单个职员（John Parker）的 DEPTNAME 不会更改该部门中其他职员的部门名。现在部门号 E11 有两个不同的部门名。产生的不一致显示在该表的更新版本中。

表 14. 更新前的不规范 EMPLOYEE_DEPARTMENT 表

EMPNO（主键）	FIRSTNAME	LASTNAME	WORKDEPT	DEPTNAME
000290	John	Parker	E11	Operations
000320	Ramlal	Mehta	E21	Software Support
000310	Maude	Setright	E11	Operations

表 15. 更新后的不规范 EMPLOYEE_DEPARTMENT 表. （表中的信息变得不一致）

EMPNO（主键）	**FIRSTNAME**	**LASTNAME**	**WORKDEPT**	**DEPTNAME**
000290	John	Parker	E11	Installation Mgmt
000320	Ramlal	Mehta	E21	Software Support
000310	Maude	Setright	E11	Operations

可创建一个含有 WORKDEPT 列和 DEPTNAME 列的新表，将该表规范化。诸如更改部门名之类的更新现在更加简单；只需要更新这个新表。

返回部门名和职员名的 SQL 查询编写起来更复杂，因为它需要连接两个表。它运行的时间可能比对单个表查询的时间长。因为 WORKDEPT 列必须出现在两个表中，所以需要更多存储空间。

下面的表被定义为规范化的结果:

表 16. 将 EMPLOYEE_DEPARTMENT 表规范化之后的 EMPLOYEE 表

EMPNO（主键）	**FIRSTNAME**	**LASTNAME**	**WORKDEPT**
000290	John	Parker	E11
000320	Ramlal	Mehta	E21
000310	Maude	Setright	E11

表 17. 将 EMPLOYEE_DEPARTMENT 表规范化之后的 DEPARTMENT 表

DEPTNO（主键）	**DEPTNAME**
E11	Operations
E21	Software Support

第四范式

若没有一行包含有关一个实体的两个或更多个独立的多值事实，则该表使用的是第四范式。

考虑下列实体：职员、技能和语言。一个职员可有几种技能，并通晓几种语言。有两个关系，一个关系是职员和技能之间的关系，另一个关系是职员和语言之间的关系。若一个表表示了这两种关系，则它未使用第四范式，如以下示例所示:

表 18. 违反第四范式的表

EMPNO（主键）	**SKILL （主键）**	**LANGUAGE （主键）**
000130	Data Modelling	英语
000130	Database Design	英语
000130	Application Design	英语
000130	Data Modelling	西班牙语
000130	Database Design	西班牙语
000130	Application Design	西班牙语

应改用两个表表示这种关系:

表 19. 符合第四范式的 EMPLOYEE_SKILL 表

EMPNO（主键）	SKILL （主键）
000130	Data Modelling
000130	Database Design
000130	Application Design

表 20. 符合第四范式的 EMPLOYEE_LANGUAGE 表

EMPNO（主键）	LANGUAGE （主键）
000130	英语
000130	西班牙语

但是，若这些属性是独立的（即，职员只将特定的语言应用于特定的技能），则不应将该表分割。

设计数据库的一个良好策略是将所有数据安排在使用第四范式的表中，然后决定该结果是否提供了一个可接受的性能级别。如果没有，则可将数据重新安排在使用第三范式的表中，然后重新评价性能。

约束

*约束*是数据库管理器强制使用的规则。

有四种类型的约束:

- *唯一约束*是这样一个规则，它禁止表的一列或多列中出现重复值。唯一键和主键是受支持的唯一约束。例如，可对供应商表中的供应商标识定义唯一约束以确保不会对两个供应商指定同一供应商标识。
- *引用约束*是关于一个或多个表中的一列或多列中的值的一种逻辑规则。例如，一组表共享关于公司的供应商的信息。供应商的名称有时可能会更改。可定义一个引用约束，声明表中的供应商的标识必须与供应商信息中的供应商标识相匹配。此约束会阻止可能导致丢失供应商信息的插入、更新或删除操作。
- *表检查约束*对添加至特定表的数据设置限制。例如，表检查约束可确保每当在包含个人信息的表中添加或更新薪水数据时，职员的薪水级别至少为 $20,000。
- *参考约束*是 SQL 编译器可以使用的规则，但这不是由数据库管理器强制执行的。

可打开或关闭引用约束和表检查约束。例如，通常在将大量数据装入数据库时关闭约束是一个好的主意。

唯一约束

*唯一约束*是这样一种规则，仅当某个键的值在表中是唯一的时候，这些值才有效。唯一约束是可选的，可在 CREATE TABLE 或 ALTER TABLE 语句中使用 PRIMARY KEY 子句或 UNIQUE 子句来定义。在唯一约束中指定的列必须定义为 NOT NULL。数据库管理器使用唯一索引在对唯一约束的各列进行更改时强制键的唯一性。

表可以有任意数目的唯一约束，最多将一个唯一约束定义为主键。对于同一组列，表不能有多个唯一约束。

引用约束的外键引用的唯一约束称为*父键*。

当在 CREATE TABLE 语句中定义唯一约束时，唯一索引是由数据库管理器自动创建的，且被指定为主索引或系统所需的唯一索引。

当在 ALTER TABLE 语句中定义唯一约束且同一组列存在索引时，该索引被指定为唯一的且是系统所需的。如果这样的索引不存在，数据库管理器会自动创建唯一索引，并将其指定为主索引或系统所需的唯一索引。

注意，定义唯一约束与创建唯一索引是有区别的。尽管都强制唯一性，但唯一索引允许可空列，且通常不能用作父键。

引用约束

*引用完整性*是数据库的一种状态，在该状态中，所有外键的所有值都有效。*外键*是表中的一列或一组列，它的值需要与其父表的行的至少一个主键或唯一键值相匹配。*引用约束*是这样一种规则，仅当满足下列其中一个条件时，外键的值才有效:

- 它们作为父键的值出现。
- 外键的某些组成部分为空。

包含父键的表称为引用约束的*父表*，包含外键的表被认为是该表的*从属表*。

引用约束是可选的，可在 CREATE TABLE 语句或 ALTER TABLE 语句中定义它。引用约束在执行 INSERT、UPDATE、DELETE、ALTER TABLE、ADD CONSTRAINT 和 SET INTEGRITY 语句时由数据库管理器强制使用。

在强制使用所有其他引用约束之前，会先强制使用带有删除或更新规则 RESTRICT 的引用约束。在大部分情况下，带有删除或更新规则 NO ACTION 的引用约束的行为类似 RESTRICT。

注意，引用约束、检查约束和触发器可以组合使用。

引用完整性规则涉及下列概念和术语:

父键 引用约束的主键或唯一键。

父行 具有至少一个从属行的行。

父表 包含引用约束的父键的表。表可在任意数目的引用约束中充当父表。在引用约束中充当父表的表还可以是引用约束中的从属表。

从属表 在其定义中包含至少一个引用约束的表。表可在任意数目的引用约束中充当从属表。在引用约束中充当从属表的表还可以是引用约束中的父表。

派生表 作为表 T 的后代的表（如果它是 T 的从属表或是 T 的从属表的后代）。

从属行 具有至少一个父行的行。

派生行 作为行 r 的后代的行（如果它是 r 的从属行或是 r 的从属行的后代）。

引用循环
它是这样的一组引用约束，该组中的每个表都是它本身的后代。

自引用表

在同一引用约束中既充当父表又充当从属表的表。该约束称为*自引用约束*。

自引用行

充当本身的父行的行。

插入规则

引用约束的插入规则为：外键的非空插入值必须与父表的父键的某些值相匹配。如果组合外键的值的任何组成部分为空，则该值为空。指定外键时，此规则是隐式的。

更新规则

引用约束的更新规则是在定义引用约束时指定的。选项有 NO ACTION 和 RESTRICT。在更新父表的某行或从属表的某行时应用更新规则。

如果是父行，更新父键的某列中的值时，下列规则适用:

- 如果从属表中的任何行与该键的原始值相匹配，在更新规则为 RESTRICT 的情况下，会拒绝更新。
- 如果在更新语句完成时从属表中的任意行没有相应的父键（排除 AFTER 触发器），当更新规则为 NO ACTION 时，会拒绝更新。

如果是从属行，当指定外键时，NO ACTION 更新规则是隐式的。NO ACTION 意味着更新语句完成时，外键的非空更新值必须与父表的父键的某些值相匹配。

如果组合外键的值的任何组成部分为空，则该值为空。

删除规则

引用约束的删除规则是在定义引用约束时指定的。选项有 NO ACTION、RESTRICT、CASCADE 或 SET NULL。仅当外键的某些列允许空值时，才能指定 SET NULL。

在删除父表的某行时应用引用约束的删除规则。更精确地说，当父表的某行是删除或传播删除操作（将在下面定义）的对象，且该行在引用约束的从属表中具有从属项时，该规则适用。考虑这样一个示例，其中 P 是父表，D 是从属表，而 p 是充当删除或传播删除操作的对象的父行。删除规则的工作方式如下:

- 对于 RESTRICT 或 NO ACTION，发生错误，且不会删除任何行。
- 对于 CASCADE，删除操作会传播至表 D 中的 p 的从属项。
- 对于 SET NULL，表 D 中的 p 的每个从属项的外键的每个可空列被设置为空。

表在其中充当父表的每个引用约束有它自己的删除规则，所有适用的删除规则被用来确定删除操作的结果。因此，在带有删除规则 RESTRICT 或 NO ACTION 的引用约束中，如果行具有从属项，则不能删除该行，或在带有删除规则 RESTRICT 或 NO ACTION 的引用约束中，删除会级联至充当从属项的任何一个后代。

从父表 P 删除一行涉及其他表，可能会影响这些表的各行:

- 如果表 D 是 P 的从属项，且删除规则为 RESTRICT 或 NO ACTION，则该操作会涉及 D，但 D 不会受该操作影响。
- 如果 D 是 P 的从属项，且删除规则为 SET NULL，则该操作涉及 D，且在执行该操作期间会更新 D 的各行。

- 如果 D 是 P 的从属项且删除规则为 CASCADE，则该操作涉及 D，且 D 的各行会在执行该操作期间删除。

  如果 D 的各行被删除，则我们说针对 P 的删除操作已传播至 D。如果 D 也是父表，则此列表中描述的操作依次适用于 D 的从属项。

涉及针对 P 的删除操作的任何表都被认为是*删除连接*至 P。因此，如果一个表是 P 的从属项，或是 P 中的删除操作级联至的表的从属项，则该表删除连接至表 P。

下列限制适用于删除连接关系:

- 如果一个表在多个表的引用循环中删除连接至它自己，则该循环不能包含删除规则 RESTRICT 或 SET NULL。
- 一个表不能既是 CASCADE 关系中的从属表（自引用或者引用另一个表）又与删除规则 RESTRICT 或 SET NULL 具有自引用关系。
- 当一个表通过多种关系（这些关系具有重叠的外键）删除连接至另一个表时，这些关系必须具有相同的删除规则，并且任何这些关系都不能为 SET NULL。
- 当一个表通过多种关系（其中一种关系是使用删除规则 SET NULL 指定的）删除连接至另一个表时，此关系的外键定义不能包含任何分布键或 MDC 键列。
- 当两个表通过 CASCADE 关系删除连接至同一个表时，这两个表之间不能互相删除连接（其中删除连接路径以删除规则 RESTRICT 或 SET NULL 结束）。

表检查约束

*表检查约束*是这样一种规则，它指定表中每行的一列或多列中允许使用的值。约束是可选的，可使用 CREATE TABLE 或 ALTER TABLE 语句定义它。指定表检查约束是通过限制格式的搜索条件完成的。其中一个限制是表 T 的表检查约束中的列名必须标识表 T 中的某列。

表可以有任意数目的表检查约束。表检查约束是通过对插入或更新的每行应用其搜索条件强制使用的。如果搜索条件的结果对任何行都是 false，则发生错误。

当在 ALTER TABLE 语句中对带有现有数据的表定义一个或多个表检查约束时，会在 ALTER TABLE 语句完成前针对新条件检查现有数据。SET INTEGRITY 语句可用来将表置于*完整性暂挂*状态，这允许 ALTER TABLE 语句继续进行而不检查数据。

参考约束

*参考约束*是 SQL 编译器可用来改善数据的访问路径的规则。参考约束不是由数据库管理器强制执行的，并且不用于数据的附加验证；它们用来提高查询性能。

使用 CREATE TABLE 或 ALTER TABLE 语句来定义引用约束或表检查约束，指定一些约束属性，这些属性确定数据库管理器是否强制执行约束以及是否将约束用于查询优化。

相关参考:

- 『Interaction of triggers and constraints』（*SQL Reference, Volume 1*）
- 『SET INTEGRITY statement』（*SQL Reference, Volume 2*）

触发器

*触发器*定义这样一组操作，这些操作是在响应对指定表的插入、更新或删除操作时执行的。执行这样的 SQL 操作时，触发器被认为是已*激活的*。

触发器是可选的，且是使用 CREATE TRIGGER 语句定义的。

可将触发器与引用约束和检查约束配合使用，以强制使用数据完整性规则。还可使用触发器来导致对其他表的更新、自动生成或变换插入或更新的行的值或调用函数以执行如发出警报之类的任务。

对于定义或强制*事务性*业务规则，触发器是非常有用的机制，这些规则涉及数据的不同状态（例如，薪水增长不能超过 10%）。

使用触发器会设置逻辑以在数据库内强制使用业务规则。这表示应用程序不负责强制使用这些规则。对所有表强制使用的集中逻辑意味着更容易维护，因为在逻辑更改时，不需要更改应用程序。

下列各项是在创建触发器时指定的:

- *主题表*指定对其定义触发器的表。
- *触发事件*定义修改主题表的特定 SQL 操作。该事件可以是插入、更新或删除操作。
- *触发器激活时间*指定触发器应在触发事件发生之前还是之后激活。

导致触发器激活的语句包括*一组受影响的行*。这些行就是正对其进行插入、更新或删除操作的主题表的行。*触发器详细程度*指定触发器的操作是对该语句执行一次，还是对每个受影响的行执行一次。

*触发操作*包括可选搜索条件和每次激活触发器时执行的一组 SQL 语句。仅当搜索条件求值为 true 时，才执行这些 SQL 语句。如果触发器激活时间是在触发事件之前，触发操作可包括选择、设置转换变量或发出 SQLstates 信号的语句。如果触发器激活时间是在触发事件之后，触发操作可包括选择、插入、更新、删除或发出 SQLstates 信号的语句。

触发操作可使用*转换变量*来引用一组受影响的行中的值。转换变量使用由指定的名称限定的主题表中的各个列名，以标识是引用旧值（更新前）还是引用新值（更新后）。在之前、插入或更新触发器中，还可使用 SET Variable 语句来更改新值。

引用一组受影响的行中的各个值的另一种方法是使用*转换表*。转换表同样使用主题表中的各个列名，但它指定一个名称，以允许将一整组受影响的行视作一个表。只能在之后触发器中使用转换表，且可对旧值和新值定义独立的转换表。

可对表、事件或激活时间的组合指定多个触发器。激活触发器的顺序与创建它们的顺序相同。因此，最新创建的触发器就是最后激活的触发器。

激活触发器可能导致*触发器级联*，这是激活一个执行 SQL 语句的触发器的结果，它会导致激活另一触发器甚至是同一触发器。触发操作还可能导致对删除应用引用完整性规则而产生的更新，从而可能导致激活更多触发器。借助触发器级联，可能会激活触发器和引用完整性删除规则链，从而由于单个 INSERT、UPDATE 或 DELETE 语句而导致对数据库的大幅度更改。

当多个触发器对同一对象执行插入、更新，或删除操作时，使用临时表避免访问冲突。但这样会对性能产生明显的影响，尤其在分区数据库环境中。

相关概念:

- 『Triggers in application development』（*Developing SQL and External Routines*）

相关任务:

- 『创建触发器』（*《管理指南：实施》*）

相关参考:

- 『Interaction of triggers and constraints』（*SQL Reference, Volume 1*）

附加数据库设计注意事项

当设计一个数据库时，重要的是考虑用户应该能够访问哪些表。通过授权，授予或撤销对表的访问权。最高权限级别是系统管理权限（SYSADM）。具有 SYSADM 权限的用户可指定其他授权，包括数据库管理员权限（DBADM）。

为了进行*审计*，可能必须在指定的时间段内记录每次对数据所做的更新。例如，您可能希望在每次职员工资改变时更新审计表。若定义了适当的触发器，则可自动更新此表。还可通过 DB2 数据库 Linux 版、UNIX 版和 Windows 版审计设施来执行审计活动。

考虑到性能方面的原因，可能只想访问选择的数据量，同时将基本数据作为*历史数据*维护。应在设计中加入对维护此历史数据的要求，如数据可以保留几个月或几年才可以清除。

您可能还想使用*总结*信息。例如，您可能有一个表，它具有所有职员的信息。但是，您希望将此信息按分部或部门划分为单独的表。在这种情况下，基于原始表中的数据的每个分部或部门的具体化查询表将非常有帮助。

在设计中还应标识隐含的*安全性*考虑事项。例如，您可能决定通过安全性表支持用户访问特定类型的数据。可定义各种数据的访问级别以及谁可访问此数据。机密数据，如职员和工资单数据，将具有严格的安全性限制。

可以创建具有相关的*结构化类型*的表。利用这种类型表，可以使用表之间定义的关系建立一个层次结构，称为*类型层次结构*。该类型层次结构由单个的根类型、超类型和子类型组成。

*引用类型*表示法是在创建类型层次结构的根类型时定义的。引用的目标始终是类型表或视图中的行。

当在"高可用性灾难恢复"（HADR）环境中工作时，有几点建议:

- HADR 环境的两个实例在硬件和软件方面应该完全相同。这意味着:
  - HADR 主数据库和备用数据库的主计算机应该完全相同。
  - 主系统和备用系统上的操作系统应该具有相同的版本（包括补丁）。（在升级期间，这可能无法做到。但是，应该使实例步调不一致的时间尽量短，以限制因存在差异而带来的任何困难。这些差异包括在故障转移期间不支持新的功能部件以及会影响常规非故障转移操作的任何问题。）
  - 在两个 HADR 实例中都必须提供 TCP/IP 接口。

- 建议使用高速高容量网络。
- 有几个应该考虑的 DB2 数据库需求。
  - 主数据库和备用数据库使用的数据库发行版应该完全相同。
  - 主 DB2 软件和备用 DB2 软件必须具有相同的位大小。(也就是说，都必须为 32 位或 64 位)。
  - 在主数据库和备用数据库上，表空间和它们相应的容器应该完全相同。对于表空间必须是对称的那些特征包括(但不仅限于)：表空间类型(DMS 或 SMS)、表空间大小、容器路径、容器大小和容器文件类型(原始设备或文件系统)。可以使用相对容器路径，在这种情况下，每个实例上的相对路径都必须相同；它们可以映射至同一绝对路径或不同的绝对路径。
  - 主数据库和备用数据库不需要具有相同的数据库路径(如使用 CREATE DATABASE 命令时声明的那样)。
  - 对备用数据库重新执行对主数据库执行的缓冲池操作，这表明主数据库与备用数据库具有相同的内存量的重要性。

第 5 章 物理数据库设计

完成逻辑数据库设计后，关于数据库和表所在的物理环境，有几个问题需要考虑。这些问题包括了解为支持和管理数据库而创建的文件，了解存储数据所需的空间大小，确定如何使用存储数据所需的表空间，以及确定用来保存数据的表的结构。本章将讨论这些问题。

数据库目录和文件

当创建一个数据库时，关于该数据库的信息（包括缺省信息）会存储在目录层次结构中。此分层目录结构的创建位置取决于您在 CREATE DATABASE 命令中提供的信息。如果您在创建数据库时未指定目录路径或驱动器的位置，则使用缺省位置。

建议您明确指出希望在何处创建数据库。

在 CREATE DATABASE 命令中指定的目录中，将创建一个使用实例名的子目录。这个子目录确保在同一目录下的不同实例中创建的数据库不会使用相同的路径。在实例名子目录下面，将创建一个名为 NODE0000 的子目录。这个子目录可以区分逻辑分区数据库环境中的数据库分区。在节点名目录下面，将创建一个名为 SQL00001 的子目录。此子目录的名称使用了数据库标记并表示正在创建的数据库。SQL00001 包含与第一个创建的数据库以及随后创建的具有更高编号（SQL00002等等）的数据库相关联的对象。这些子目录可以区分在 CREATE DATABASE 命令中指定目录下的实例中创建的数据库。

目录结构如下所示:

```
<your_directory>/<your_instance>/NODE0000/SQL00001/
```

数据库目录中包含下列作为 CREATE DATABASE 命令的一部分进行创建的文件。

- 文件 SQLBP.1 和 SQLBP.2 中包含缓冲池信息。这两个文件中具有相同的副本，从而提供备份。
- 文件 SQLSPCS.1 和 SQLSPCS.2 中包含表空间信息。这两个文件中具有相同的副本，从而提供备份。
- 文件 SQLSGF.1 和 SQLSGF.2 包含与数据库的自动存储器有关的存储器路径信息。这两个文件中具有相同的副本，从而提供备份。
- SQLDBCON 文件中包含数据库配置信息。切勿编辑此文件。要更改配置参数，请使用控制中心或命令行语句 UPDATE DATABASE CONFIGURATION 和 RESET DATABASE CONFIGURATION。
- DB2RHIST.ASC 历史记录文件及其备份 DB2RHIST.BAK 中包含关于备份、复原、表装入、表重组、表空间改变和其他数据库更改的历史记录信息。

  DB2TSCHNG.HIS 文件中包含日志文件级别的表空间更改的历史记录。对于每个日志文件，DB2TSCHG.HIS 中包含有助于标识日志文件影响哪些表空间的信息。表空间恢复使用此文件中的信息来确定在进行表空间恢复期间要处理哪些日志文件。可以在文本编辑器中检查这两个历史记录文件中的内容。
- 日志控制文件 SQLOGCTL.LFH 和 SQLOGMIR.LFH 中包含有关活动日志的信息。

恢复处理过程使用此文件中的信息来确定要在日志中后退多远来开始恢复。SQLOGDIR 子目录包含实际的日志文件。

注: 您应确保不要将日志子目录映射到用于存储数据的磁盘。这样，在磁盘发生问题时，只会影响到数据或日志，而不会同时影响这两者。由于日志文件与数据库容器不会争用同一磁盘磁头的移动，因此这可提供很多性能方面的好处。要更改日志子目录的位置，请更改 *newlogpath* 数据库配置参数。

- SQLINSLK 文件用于确保一个数据库只能由数据库管理器的一个实例使用。

在创建数据库的同时，还创建了详细死锁事件监视器。详细的死锁事件监视器文件存储在目录节点的数据库目录中。当事件监视器达到它要输出的最大文件数时，它将取消激活，并且将把一条消息写入通知日志中。这样可防止事件监视器消耗过多的磁盘空间。除去不再需要的输出文件将允许在下一次数据库激活时再次激活事件监视器。

SMS 数据库目录的其他信息

SQLT* 子目录中包含运作数据库所需的缺省“系统管理的空间”（SMS）表空间。将创建三个缺省表空间:

- SQLT0000.0 子目录中包含带有系统目录表的目录表空间。
- SQLT0001.0 子目录中包含缺省临时表空间。
- SQLT0002.0 子目录中包含缺省用户数据表空间。

每个子目录或容器中都会创建一个名为 SQLTAG.NAM 的文件。这个文件可以标记正在使用中的子目录，因此在以后创建其他表空间时，不会尝试使用这些子目录。

此外，名为 SQL*.DAT 的文件中还存储有关子目录或容器包含的每个表的信息。星号（*）将被唯一的一组数字取代，用来识别每个表。对于每个 SQL*.DAT 文件，可能有一个或多个下列文件，这取决于表类型、表的重组状态或者表是否存在索引、LOB 或 LONG 字段:

- SQL*.BKM（如果是 MDC 表，则包含块分配信息）
- SQL*.LF（包含 LONG VARCHAR 或 LONG VARGRAPHIC 数据）
- SQL*.LB（包含 BLOB、CLOB 或 DBCLOB 数据）
- SQL*.XDA（包含 XML 数据）
- SQL*.LBA（包含关于 SQL*.LB 文件的分配和可用空间信息）
- SQL*.INX（包含索引表数据）
- SQL*.IN1（包含索引表数据）
- SQL*.DTR（包含用于重组 SQL*.DAT 文件的临时数据）
- SQL*.LFR（包含用于重组 SQL*.LF 文件的临时数据）
- SQL*.RLB（包含用于重组 SQL*.LB 文件的临时数据）
- SQL*.RBA（包含用于重组 SQL*.LBA 文件的临时数据）

相关概念:

- 第 131 页的『SMS 和 DMS 表空间的比较』
- 第 110 页的『数据库管理的空间』
- 第 114 页的『DMS 设备注意事项』
- 第 113 页的『DMS 表空间』

- 第 109 页的『SMS 表空间』
- 『了解恢复历史记录文件』（*《数据恢复及高可用性指南与参考》*）

相关参考:

- 『CREATE DATABASE command』（*Command Reference*）

数据库对象的空间需求

数据库对象的大小估计不可能做到很精确。磁盘碎片、可用空间以及使用变长列所造成的开销都使大小估计变得十分困难，因为可能的列类型和行长度的范围实在是太广了。在最初估计数据库大小后，创建一个测试数据库，并用有代表性的数据填充它。

在“控制中心”中，可访问许多专门辅助您确定各种数据库对象的大小需求的实用程序:

- 可选择一个对象，然后使用“估计大小”实用程序。此实用程序可告诉您现有对象（例如，表）的当前大小。在更改该对象之后，该实用程序可以对该对象计算出新的估计值。该实用程序可以帮助您估计存储器需求（将未来的增长考虑在内）。它提供对象可能的大小范围：最小大小（基于当前值）和可能的最大大小。
- 可通过使用“显示相关内容”窗口来确定对象之间的关系。
- 可选择实例中的任何一个数据库对象，然后请求“生成 DDL”。此功能使用 **db2look** 实用程序来生成数据库的数据定义语句。

在这两种情况下，可使用“显示 SQL”或“显示命令”按钮。可以将生成的 SQL 语句或命令保存在脚本文件中，以便稍后使用。所有这些实用程序都有联机帮助可向您提供帮助。

在制订物理数据库计划时，应考虑这些实用程序。

估计数据库的大小时，必须考虑下列各项的影响:

- 系统目录表
- 用户表数据
- 长型字段数据
- 大对象（LOB）数据
- 索引空间
- 日志文件空间
- 临时工作空间

不讨论与下列各项相关的空间需求:

- 本地数据库目录文件
- 系统数据库目录文件
- 操作系统所需的文件管理开销，包括:
  - 文件块大小
  - 目录控制空间

相关概念:

- 第 73 页的『索引的空间需求』

- 第 72 页的『大对象数据的空间需求』
- 第 75 页的『日志文件的空间需求』
- 第 72 页的『长型字段数据的空间需求』
- 第 70 页的『系统目录表的空间需求』
- 第 76 页的『临时表的空间需求』
- 第 70 页的『用户表数据的空间需求』

相关参考:

- 『db2look - DB2 statistics and DDL extraction tool command』(*Command Reference*)

系统目录表的空间需求

创建数据库时，会创建系统目录表。当将数据库对象和特权添加至该数据库时，这些系统表将增大。最初，这些系统表使用大约 3.5 MB 的磁盘空间。

为目录表分配的空间容量取决于表空间的类型和包含这些目录表的表空间的扩展数据块大小。例如，若使用扩展数据块大小为 32 的 DMS 表空间，则最初会分配给目录表空间 20 MB 的空间。

注: 对于含多个分区的数据库，目录表只位于发出 CREATE DATABASE 命令的数据库分区上。目录表的磁盘空间仅对于该数据库分区才是必需的。

相关概念:

- 第 69 页的『数据库对象的空间需求』
- 『系统目录表』(*《管理指南：实施》*)

用户表数据的空间需求

在缺省情况下，表数据存储在 4KB 页上。每一页(不管页大小如何)都包含 68 个字节的数据库管理器开销。这将保留 4028 个字节来存放用户数据(或行)，虽然 4 KB 页上的行的长度不能超过 4005 个字节。一行不会横跨多页。当使用 4KB 的页大小时，最多可有 500 列。

表数据页不包含用 LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB 或 DBCLOB 数据类型定义的列的数据。但是，一个表数据页中的行的确包含这些列的描述符。

通常，以“最先合适”顺序将行插入到常规表中。(使用可用空间映射)搜索文件，查找大小足以存放该新行的第一个可用空间。当更新一行时，除非该页上所剩的空间不足以包含它，否则将对它进行原地更新。若所剩空间不足以包含新行，则在原始行位置创建一个记录，以指向更新后的行在表文件中的新位置。

若调用 ALTER TABLE APPEND ON 语句，则将一直追加数据，且不保留关于数据页上任何可用空间的信息。

如果对表定义了集群索引，则 DB2 数据库 Linux 版、UNIX 版和 Windows 版将尝试根据该集群索引的键顺序以物理方式建立数据的集群。当将一行插入该表中时，DB2 将首先在集群索引中查找它的键值。如果找到了键值，DB2 就会尝试将记录插入到该键所

指向的数据页上；如果找不到键值，则将使用下一个更大的键值，以便将记录插入到包含具有下一个更大键值的记录的数据页上。如果该表中“目标”页上的空间不足，则将使用可用空间映射来搜索邻近页以找到空间。随着时间的推移，当数据页上的空间被彻底用完时，记录将被放置在离该表中的“目标”页越来越远的位置。然后，表数据将被认为是非集群的，并且可以使用表重组来复原集群顺序。

如果表是多维集群（MDC）表，则 DB2 将保证始终根据一个或多个已定义的维或集群索引以物理方式集群记录。当 MDC 表是使用特定维数来定义时，将对每个维创建块索引，并且将创建组合块索引，它将单元格（维值的唯一组合）映射至块。此组合块索引用来确定特定记录属于哪个单元格以及表中的哪些块或扩展数据块包含属于该单元格的记录。因此，当插入记录时，DB2 将搜索组合块索引以找到包含具有相同维值的记录的块列表，并且将搜索空间的范围仅限于这些块。如果单元格尚不存在，或者如果单元格的现有块中空间不足，则将把另一个块分配给该单元格，并且将记录插入其中。仍然在块中使用可用空间映射来快速找到块中的可用空间。

对于数据库中的每个用户表，可以通过如下计算公式来估计 4KB 页数:

```
ROUND DOWN(4028/(average row size + 10)) = records_per_page
```

然后，将结果插入:

```
(number_of_records/records_per_page) * 1.1 = number_of_pages
```

其中，平均行大小是平均列大小的总和，而因子“1.1”表示开销。

注: 此公式只是提供一个估计值。若记录长度因碎片和溢出记录而改变，则估计的准确性将降低。

也可以选择创建具有 8 KB、16 KB 或 32 KB 页大小的缓冲池或表空间。在特定大小的表空间中创建的所有表都将具有匹配的页大小。假设使用 32 KB 的页大小，则单个表或索引对象的最大大小可达 512 GB。当使用 8 KB、16 KB 或 32 KB 的页大小时，最多可有 1012 列。对于 4 KB 的页大小，最大列数为 500。最大行宽也随页大小的不同而不同:

- 当页大小是 4 KB 时，行长度最大可为 4005 个字节。
- 当页大小是 8 KB 时，行长度最大可为 8101 个字节。
- 当页大小是 16 KB 时，行长度最大可为 16293 个字节。
- 当页大小是 32 KB 时，行长度最大可为 32677 个字节。

更大的页大小有助于减小任何索引中的级别数。若使用执行随机行读写的 OLTP（联机事务处理）应用程序，则小一点的页大小会更好，这样，因意外的行而浪费的缓冲区空间更少。若使用一次访问大量连续行的 DSS（决策支持系统）应用程序，则大一点的页大小会更好，这样可以减少读取特定行数所需的 I/O 请求数。当行大小小于页大小除以 255 的值时，会发生异常。在这种情况下，每个页上都有浪费的空间。（每页最多只能有 255 行。）为减少浪费的空间，小一点的页大小可能合适。

不能将备份复原为另一种页大小。

不能导入超过 755 列的 IXF 数据文件。

已声明临时表只能采用它们自己的“用户临时”表空间类型创建。没有缺省用户临时表空间。临时表不能含有 LONG 数据。当应用程序与数据库断开连接时，这些表被隐式废弃，估计它们的空间需求时应将这一点考虑在内。

相关概念:

• 第 69 页的『数据库对象的空间需求』

长型字段数据的空间需求

长型字段数据存储在单独的表对象中，它的结构与其他数据类型的存储空间不同。

数据存储在一些大小为 32 KB 的区域中，这些区域被分成大小为 512 个字节的“2 的幂”倍的段。(因此，这些段可以是 512 个字节、1024 个字节和 2048 个字节，以此类推，直至 32768 个字节。)

长型字段数据类型（LONG VARCHAR 或 LONG VARGRAPHIC）以使可用空间易于收回的方式存储。有关分配和可用空间的信息存储在 4KB 分配页中，它在整个对象中不经常出现。

对象中未使用的空间量取决于长型字段数据的大小以及此大小是否在该数据的所有出现之处都是不变的。对于大于 255 个字节的数据条目，这个未使用的空间最大可为该长型字段数据大小的 50%。

若该字符数据的长度小于页大小，且它适合含有该数据其余部分的记录，则应该使用 CHAR、GRAPHIC、VARCHAR 或 VARGRAPHIC 数据类型，而不要使用 LONG VARCHAR 或 LONG VARGRAPHIC。

相关概念:

• 第 69 页的『数据库对象的空间需求』

大对象数据的空间需求

大对象（LOB）数据存储在两个单独的表对象中，这两个对象的结构与其他数据类型的存储空间不同。

要估计 LOB 数据所需的空间，需要考虑用来存储使用这些数据类型定义的数据的两个表对象:

- **LOB 数据对象**

  数据存储在一些大小为 64 MB 的区域中，这些区域被分成大小为 1024 字节的“2 的幂”倍的段。(因此，这些段可以是 1024 个字节、2048 个字节和 4096 个字节，以此类推，直至 64MB。)

  要减少 LOB 数据所用的磁盘空间量，可在 CREATE TABLE 和 ALTER TABLE 语句上的 *LOB 选项*子句上使用 COMPACT 参数。COMPACT 选项将所需的磁盘空间量减至最小，方法是将 LOB 数据分成更小的段。此过程不涉及数据压缩，只是使用最接近 1 KB 边界的最小空间量。使用 COMPACT 选项可能导致在追加 LOB 值时性能下降。

  包含在 LOB 数据对象中的可用空间量将受到更新和删除活动量以及要插入的 LOB 值的大小的影响。

- **LOB 分配对象**

有关分配和可用空间的信息存储在与实际数据分离的 4 KB 分配页中。这些 4KB 页的数目取决于数据量，包括为大对象数据分配的未使用空间。开销的计算如下：每 64 GB 一个 4 KB 页加上每 8 MB 一个 4KB 页。

若该字符数据的长度小于页大小，且它适合含有该数据其余部分的记录，则应该使用 CHAR、GRAPHIC、VARCHAR 或 VARGRAPHIC 数据类型，而不要使用 BLOB、CLOB 或 DBCLOB。

相关概念:

- 第 69 页的『数据库对象的空间需求』

相关参考:

- 『Large objects (LOBs)』（*SQL Reference, Volume 1*）

索引的空间需求

对于每个索引，可以按如下公式估计所需的空间:

```
(平均索引键大小 + 9) * 行数 * 2
```

其中:

- “平均索引键大小”是索引键中每列的字节计数。（估计 VARCHAR 和 VARGRAPHIC 列的平均列大小时，使用当前数据大小的平均值加上两个字节。不要使用最大声明大小。）
- 因子“2”表示开销，如非叶子页和可用空间。

注:

1. 对于允许 NULL 的每个列，为空指示符添加一个额外的字节。
2. 对于在内部为多维集群（MDC）表创建的块索引，“行数”将被替换为“块数”。

在版本 8 之前创建的索引（1 类索引）与在版本 8（2 类索引）及其后续版本创建的索引不同。要了解一个表存在什么类型的索引，可使用 INSPECT 命令。要将 1 类索引转换为 2 类索引，可使用 REORG INDEXES 命令。

在使用 REORG INDEXES 命令时，确保存储索引的表空间中有足够的可用空间。可用空间的大小应等于索引的当前大小。如果选择使用 ALLOW WRITE ACCESS 选项来重组索引，则可能需要其他空间。其他空间用于存储重组索引期间影响索引的活动的日志。

创建索引时，临时空间是必需的。在创建索引期间所需的最大临时空间可以按如下公式估计:

```
(平均索引键大小 + 9) * 行数 * 3.2
```

其中，因子“3.2”表示索引开销以及索引创建期间进行排序所需的空间。

注: 对于非唯一索引，存储重复键条目只需五个字节。上面显示的估计是假定没有重复的条目。因此以上公式可能会过多地估计存储索引所需的空间。

可使用下面两个公式来估计每个叶子页的键数（第二个公式提供更准确的估计）。这些估计的准确度很大程度上取决于平均值反映实际数据的准确程度。

注: 对于 SMS 表空间，叶子页所需的最小空间为 12 KB。对于 DMS 表空间，最小值是一个扩展数据块。

- 每个叶子页的平均键数的粗略估计是:

```
(.9 * (U - (M*2))) * (D + 1)
----------------------------
      K + 7 + (5 * D)
```

  其中:

  – U（一页上的可用空间）大约等于页大小减 100。对于 4096 的页大小，U 是 3996。
  – M = U / (9 + *最小键大小*)
  – D = 每个键值的平均重复项数目
  – K = *平均键大小*

  记住，*最小键大小*和*平均键大小*必须有一个额外字节，表示每个可空键部分；还必须有两个额外字节，表示每个变长键部分的长度。

  若存在包含列，则在计算*最小键大小*和*平均键大小*时应将它们考虑在内。

  若在创建索引期间指定了非缺省值 10% 的一个可用百分比值，则可以使用任何 (100 - pctfree)/100 值替换 .9。

- 每个叶子页的平均键数的更准确估计是:

```
L = 叶子页数 = X / (叶子页上的平均键数)
```

  其中，X 是表中的总行数。

  可按如下方法估算索引的原始大小:

```
(L + 2L/(叶子页上的平均键数)) * 页大小
```

  对于 DMS 表空间，将一个表上所有索引的大小加在一起，然后四舍五入为该索引所在表空间的扩展数据块大小的一个倍数。

  应该为 INSERT/UPDATE 活动所引起的索引增长提供其他空间，这种增长可能导致分页。

  使用以下计算方法来获得更精确的原始索引大小的估算值，以及该索引中级别数的估算值。（若索引定义中使用包含列，可能要引起特别注意。）每个非叶子页的平均键数大约是:

```
     (.9 * (U - (M*2))) * (D + 1)
----------------------------
      K + 13 + (9 * D)
```

  其中:

  – U（一页上的可用空间）大约等于页大小减 100。对于 4096 的页大小，U 是 3996。
  – D 是在非叶子页上每个键值的重复值的平均数目（这将比在叶子页上的小很多，您可能想将该值设置为 0 以便简化计算）。
  – M = U / (9 + 非叶子页的*最小键大小*)
  – K = 非叶子页的*平均键大小*

只要没有包含列，非叶子页与叶子页的*最小键大小*和*平均键大小*将是相同的。包含列不存储在非叶子页上。

除非 (100 - pctfree)/100 大于 .9，否则不应使用它来替换 .9，因为在创建索引期间会在非叶子页上留下最多 10% 的可用空间。

可用如下所示的方法估算非叶子页数:

```
    if L > 1 then {P++; Z++}
 While (Y > 1)
 {
    P = P + Y
    Y = Y / N
   Z++
 }
```

其中:

– P 是页数（最初为 0）。

– L 是叶子页数。

– N 是每个非叶子页的键数。

– Y = L / N

– Z 是索引树中的级别数（最初为 1）。

总页数是:

```
T = (L + P + 2) * 1.0002
```

附加的 0.02% 表示开销，包括空间映射页。

创建索引所需的空间容量估算为:

```
T * 页大小
```

相关概念:

- 『Indexes』（*SQL Reference, Volume 1*）
- 『索引清除和维护』（*《性能指南》*）
- 第 69 页的『数据库对象的空间需求』

日志文件的空间需求

日志控制文件需要 32 KB 的空间。

至少还需要足够的空间以进行活动日志配置，可进行如下计算

```
(logprimary + logsecond) * (logfilsiz + 2 ) * 4096
```

其中:

- *logprimary* 是在数据库配置文件中定义的主日志文件数
- *logsecond* 是在数据库配置文件中定义的辅助日志文件的数目；在此计算中，不能将 *logsecond* 设置为 *-1*。（当 *logsecond* 设置为 -1 时，您是在请求一个无限的活动日志空间。）
- *logfilsiz* 是在数据库配置文件中定义的每个日志文件中的页数
- 2 是每个日志文件所需的标题页的数目

- 4096 是一页中的字节数

如果已对数据库启用了循环日志记录，则此公式的结果将提供足够的磁盘空间。

若允许对该数据库执行前滚恢复，应该考虑特殊的日志空间需求：

- 当启用 *logretain* 配置参数时，日志文件将被归档在日志路径目录中。除非将日志文件移至另一个位置，否则，联机磁盘空间最终将会填满。
- 当启用 *userexit* 配置参数时，用户出口程序会将归档的日志文件移至另一个位置。要允许下列情况，附加的日志空间仍是必需的：
  - 等待用户出口程序移动的联机归档日志
  - 格式化新的日志文件，以供将来使用

如果已对数据库启用无限日志记录（即，将 *logsecond* 设置为 *-1*），则必须将 *logarchmeth1* 配置参数设置为除 LOGRETAIN 的 OFF 之外的值，以便启用归档日志记录。DB2 数据库 Linux 版、UNIX 版和 Windows 版将在日志路径中至少保留 *logprimary* 指定的数目的活动日志文件，所以上面公式中 *logsecond* 的值不应使用 -1。确保提供额外磁盘空间以允许归档日志文件导致的延迟。

如果在镜像日志路径，则需要将估计的日志文件空间需求加倍。

相关概念:

- 第 69 页的『数据库对象的空间需求』
- 『日志镜像』（*《数据恢复及高可用性指南与参考》*）
- 『了解恢复日志』（*《数据恢复及高可用性指南与参考》*）

相关参考:

- 『mirrorlogpath － 镜像日志路径配置参数』（*《性能指南》*）
- 『logprimary － 主日志文件数配置参数』（*《性能指南》*）
- 『logsecond － 辅助日志文件数配置参数』（*《性能指南》*）
- 『logfilsiz － 日志文件大小配置参数』（*《性能指南》*）

临时表的空间需求

某些 SQL 语句需要临时表来进行处理（如使用一个工作文件来进行不能在内存中执行的排序）。这些临时表需要磁盘空间；所需的空间量取决于查询的大小、数目和属性以及返回的表的大小。您的工作环境是独特的，这使得很难估计临时表的空间需求。例如，由于各种系统临时表的生命周期更长，为系统临时表空间分配的空间比实际在使用的表空间更多。使用 DB2_SMS_TRUNC_TMPTABLE_THRESH 时就会发生这种情况。

可以使用数据库系统监视器和查询表空间 API 来跟踪在正常操作期间所用的工作空间量。

可以使用 DB2_OPT_MAX_TEMP_SIZE 注册表变量来限制查询所使用的临时表空间大小。

相关概念:

- 第 69 页的『数据库对象的空间需求』

相关参考:

- 『sqlbmtsq API - Get the query data for all table spaces』(*Administrative API Reference*)
- 『查询编译器变量』(*《性能指南》*)

XML 存储器对象概述

DB2 表可以将结构良好的 XML 文档存储在 XML 列中，该 XML 列旁边是包含关系数据的列。与 LONG VARCHAR 和 LOB 数据的存储位置不同于表的其他内容的存储位置这种情况相似，DB2 将 XML 数据(包含在类型为 XML 的表列中的数据)存储在辅助 XML 存储器对象中。存储在系统管理的空间中时，与 XML 存储器对象关联的文件具有文件类型扩展名 .xda。

XML 存储器对象与它们的父表对象分开，但依赖于其父表对象。对于存储在 XML 表列的行中的每个 XML 值，DB2 都维护一条称为 XML 数据说明符(XDS)的记录，该记录指定从关联的 XML 存储器对象中的何处检索存储在磁盘上的 XML 数据。

最多可以将大小为 2 吉字节的 XML 文档存储在数据库中。因为 XML 数据可能非常大，所以可能要与其他数据的缓冲活动分开单独监视 XML 数据的缓冲活动。提供了一些监视元素来帮助您监视 XML 存储器对象的缓冲池活动。

相关概念:

- 『用于本机 XML 数据存储性能的数据库管理的表空间的首选项』(*《性能指南》*)
- 第 77 页的『XML 文档的存储器需求准则』
- 『本机 XML 数据存储概述』(*《XML 指南》*)
- 『XML 数据说明符』(*《数据移动指南和参考》*)

相关参考:

- 『缓冲池活动监控器元素』(*《系统监视器指南和参考》*)

XML 文档的存储器需求准则

XML 文档在 DB2 数据库中所占用的空间大小由原始格式的文档的最初大小和一些其他属性确定。以下列表包含这些属性中最重要的一些属性:

文档结构
: 包含复杂标记的 XML 文档需要的存储器空间比具有简单标记的文档所需的空间要大。例如，如果一个 XML 文档具有许多嵌套元素，每个嵌套元素包含少量文本或具有简短的属性值，则该文档占用的存储器空间比主要由文本内容组成的 XML 文档要多。

节点名
: 元素名称、属性名称、名称空间前缀以及类似的非内容数据的长度也影响存储器的大小。压缩原始格式超过 4 个字节的任何这种类型的信息单元以进行存储，将使得存储器效率比使用较长的节点名要高。

属性数与元素数之比
: 通常，每个元素使用的属性越多，XML 文档所需的存储器空间大小就越小。

文档代码页

如果 XML 文档所使用的编码使得每个字符要使用多个字节，则该 XML 文档占用的存储器空间大小比使用单字节字符集的文档要大。

文档验证

在针对 XML 模式验证了 XML 文档之后，将注释该文档。验证之后增加类型信息将导致存储器需求增大。

相关概念:

- 第 77 页的『XML 存储器对象概述』

数据库分区组

数据库分区组是定义为属于某个数据库的一个或多个数据库分区的集合。想要为数据库创建表时，首先创建用来存储表空间的数据库分区组，然后创建用来存储表的表空间。

可以在数据库中定义一个或多个数据库分区组成的命名子集。您定义的每个子集称为*数据库分区组*。包含多个数据库分区的每个子集称为*多分区数据库分区组*。多分区数据库分区组只能使用属于相同实例的数据库分区定义。数据库分区组可以只包括一个数据库分区，也可以包括数据库中的所有数据库分区。

图 22 给出了一个含五个数据库分区的数据库示例，在这个示例中:

- 数据库分区组横跨除一个数据库分区外的所有其他分区（数据库分区组 1）。
- 数据库分区组包含一个数据库分区（数据库分区组 2）。
- 数据库分区组包含两个数据库分区（数据库分区组 3）。
- 数据库分区组 2 中的数据库分区与数据库分区组 1 共享并重叠。
- 数据库分区组 3 中有一个数据库分区与数据库分区组 1 共享并重叠。

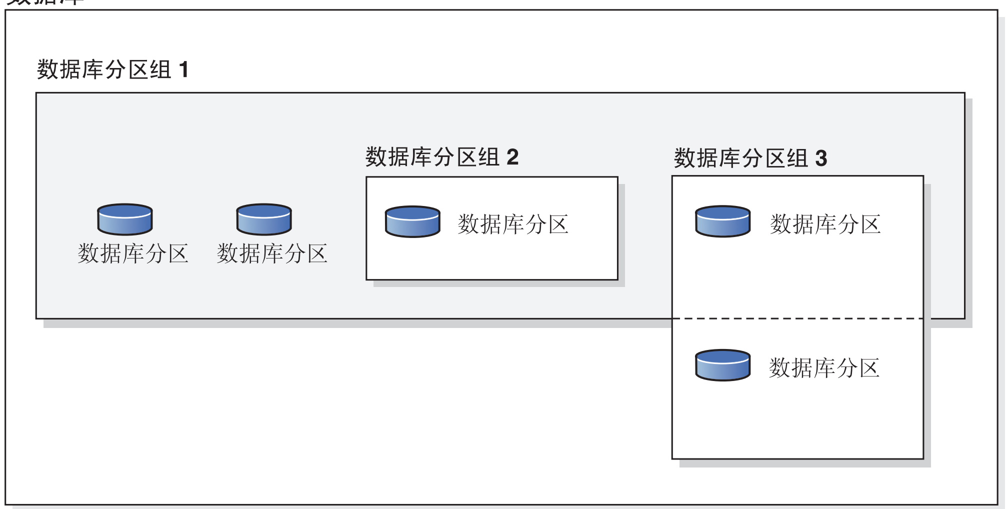


图 22. 数据库中的数据库分区组

使用 CREATE DATABASE PARTITION GROUP 语句创建新的数据库分区组。可以使用 ALTER DATABASE PARTITION GROUP 语句修改它。数据分布在数据库分区组

中的所有数据库分区上，并且可以从数据库分区组添加或删除一个或多个数据库分区。如果正在使用多分区数据库分区组，则必须查看几个数据库分区组设计注意事项。

作为数据库系统配置一部分的每个数据库分区，必须已在名为 db2nodes.cfg 的*数据库分区配置文件*中定义。数据库分区组可以只包括一个数据库分区，也可以包括为该数据库系统定义的所有数据库分区。

创建或修改数据库分区组时，*分发映射*与它是相关联的。数据库管理器将分发映射和*分布键*及散列法联合使用来确定数据库分区组中的哪个数据库分区将存储给定的数据行。

在非分区数据库中，不需要任何分布键或分发映射。数据库分区是数据库的一部分，与用户数据、索引、配置文件和事务日志相辅相成。创建数据库时创建的缺省数据库分区组将由数据库管理器使用。IBMCATGROUP 是包含系统目录的表空间的缺省数据库分区组。IBMTEMPGROUP 是系统临时表空间的缺省数据库分区组。IBMDEFAULTGROUP 是包含用户定义的表的表空间的缺省数据库分区组（可选择在其中放置用户定义的表）。已声明临时表的用户临时表空间可在 IBMDEFAULTGROUP 或用户创建的任何数据库分区组中创建，但不能在 IBMTEMPGROUP 中创建。

处理数据库分区组时，您可以:
- 创建数据库分区组。
- 更改与数据库分区组相关联的注释。
- 将数据库分区添加至数据库分区组。
- 从数据库分区组中删除数据库分区。
- 在数据库分区组中再分发表数据。

相关概念:

- 第 79 页的『数据库分区组设计』
- 第 81 页的『分布键』
- 第 80 页的『分发映射』

相关参考:

- 『ALTER DATABASE PARTITION GROUP statement』（*SQL Reference, Volume 2*）
- 『CREATE DATABASE PARTITION GROUP statement』（*SQL Reference, Volume 2*）

数据库分区组设计

若使用单分区数据库，则不考虑数据库分区组设计。

“DB2 设计顾问程序”是一个可以用来建议数据库分区组的工具。可以从控制中心来访问“DB2 设计顾问程序”，也可以从命令行处理器中使用 db2advis 来访问“DB2 设计顾问程序”。

若使用多分区数据库分区组，应考虑下列设计要点:
- 在多分区数据库分区组中，若唯一索引是该分布键的超集，则只能创建唯一索引。

- 根据该数据库中的数据库分区的数目，可能有一个或多个单一分区数据库分区组，且存在一个或多个多分区数据库分区组。
- 必须给每个数据库分区指定唯一的编号。在一个或多个数据库分区组中，可能会发现相同的数据库分区。
- 要确保快速恢复包含系统目录表的数据库分区，则须避免将用户表放在同一个数据库分区上。这可通过将用户表放在不包括 IBMCATGROUP 数据库分区组中的那些数据库分区的数据库分区组中来实现。

除非要与一个更大的表*并置*，否则应该将小表放在单一分区数据库分区组中。并置是将不同表中包含相关数据的行放置在同一个数据库分区中。并置的表允许 DB2 数据库 Linux 版、UNIX 版和 Windows 版利用更有效的连接策略。并置的表可以位于单一分区数据库分区组中。若表位于多分区数据库分区组中，且在分布键中具有相同数目的列，对应列的数据类型是兼容的，则将这些表视为并置。在并置的表中具有相同分布键值的行被放置在同一个数据库分区上。这些表可以位于相同数据库分区组中的单独表空间中，且仍被视为是并置的。

应该避免将中等大小的表扩展到太多数据库分区上。例如，100 MB 的表在 16 个分区的数据库分区组中可能比在 32 个分区的数据库分区组中执行得更好。

可以使用数据库分区组将联机事务处理（OLTP）表与决策支持（DSS）表分开，以确保不会对 OLTP 事务的性能产生负面影响。

相关概念:

- 第 78 页的『数据库分区组』
- 第 83 页的『数据库分区的兼容性』
- 第 81 页的『分布键』
- 第 80 页的『分发映射』
- 第 101 页的『复制具体化查询表』
- 第 83 页的『表并置』

相关参考:

- 『db2advis - DB2 design advisor command』（*Command Reference*）

分发映射

在分区数据库环境中，数据库管理器必须知道到哪里去查找所需的数据。数据库管理器使用一个称为*分发映射*的映射来查找数据。

分发映射是一个内部生成的数组，对于多分区数据库分区组，它包含 4096 个条目，对于单一分区数据库分区组，只包含一个条目。对于单一分区数据库分区组，分发映射只有一个条目，该条目包含存储数据库表的所有行的数据库分区的编号。对于多分区数据库分区组，数据库分区组的编号指定方式使得能够一个接一个地使用每个数据库分区，以确保整个映射分发均匀。正如使用网格将城市地图划分为区一样，数据库管理器使用*分布键*来确定存储数据的位置（数据库分区）。

例如，假定您将一个数据库创建在四个数据库分区（编号为 0 到 3）上。此数据库的 IBMDEFAULTGROUP 数据库分区组的分发映射将是:

```
0 1 2 3 0 1 2 ...
```

若已使用数据库分区 1 和 2 在该数据库中创建了一个数据库分区组，则该数据库分区组的分发映射将是:

```
1 2 1 2 1 2 1 ...
```

若要装入到数据库的一个表的分布键是一个可能在范围 1 至 500000 之间的整数，则会将分布键散列至 0 至 4095 之间的一个编号。将该编号用作分发映射中的索引，以选择用于该行的数据库分区。

图 23 显示如何将具有分布键值 (c1，c2，c3) 的行映射至编号 2，然后引用数据库分区 n5。

分区键

Row: (... c1, c2, c3, ...)

分区映射

n0	n2	n5	n0	n6	. . .
0	1	2	3	4	. . . 4095

分区号

图 23. 使用分发映射的数据分发

分发映射可以灵活地控制将数据存储在多分区数据库中的哪个位置。若需要更改数据库中各数据库分区上的数据分发，可以使用数据再分发实用程序。此实用程序允许重新平衡或调整数据分发的偏差。

可以使用 **sqlugtpi API – 获取表分发信息**来获取您可以查看的分发映射的副本。

相关概念:

- 第 79 页的『数据库分区组设计』
- 第 78 页的『数据库分区组』
- 第 81 页的『分布键』

相关参考:

- 『sqlugtpi API - Get table distribution information』（*Administrative API Reference*）

分布键

*分布键*是一列（或一组列），用来确定存储特定数据行的数据库分区。分布键是使用 CREATE TABLE 语句在表上定义的。如果没有为分布在数据库分区组中的多个数据库分区中的表空间中的表定义分布键，在缺省情况下将会根据主键的第一列创建分布键。若未指定主键，则缺省分布键是在该表中定义的第一个非长型字段列。（*长型*包括

所有长型数据类型和所有大对象（LOB）数据类型）。如果正在与单一分区数据库分区组相关的表空间中创建表，且您希望有分布键，则必须显式定义该分布键。缺省情况下，不会创建它。

若没有列满足缺省分布键的需求，则会创建不带分布键的表。仅允许没有分布键的表存在于单一分区数据库分区组中。以后可以使用 ALTER TABLE 语句来添加或删除分布键。仅可对其表空间与单一分区数据库分区组相关的表改变分布键。

选择好的分布键很重要。应考虑下列各项:

- 如何访问表
- 查询工作负载的性质
- 数据库系统所使用的连接策略

若并置不是主要的注意事项，则好的表分布键就是将数据均匀分布在数据库分区组中的所有数据库分区上的分布键。与数据库分区组相关联的表空间中每个表的分布键确定这些表是否是并置的。下列情况中，表被认为是并置的:

- 表被放置在同一数据库分区组中的表空间内
- 每个表中的分布键具有相同数量的列
- 对应列的数据类型是分区兼容的

这些特征确保并置的表中具有相同分布键值的那些行位于同一个数据库分区上。

不适当的分布键会导致数据分布不均匀。不应选择数据分布不均匀的列和含有少数单值的列作为分布键。单值的数目必须足够大，才能确保将行均匀分布在数据库分区组中的所有数据库分区上。应用分发算法的成本与分布键的大小是成正比的。分布键不能超过 16 列，而且列越少，性能越好。不应将不需要的列包括在分布键中。

当定义分布键时，应该考虑以下几点:

- 不支持创建只包含长型数据类型（LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB 或 DBCLOB）的多分区表。
- 不能改变分布键定义。
- 分布键应该包括最频繁连接的列。
- 分布键应该由经常参与 GROUP BY 子句的列组成。
- 任何唯一键或主键必须包含所有分布键列。
- 在联机事务处理（OLTP）环境中，分布键中的所有列都应该使用带常量或主变量的等于（=）谓词来参与该事务。例如，假定有一个在事务中经常使用的职员号 *emp_no*，如:

  ```
  UPDATE emp_table SET ... WHERE
  emp_no = host-variable
  ```

  在此情况下，EMP_NO 列对于 EMP_TABLE 而言是一个不错的单列分布键。

*数据库分区*是确定表中每一行的位置的方法。该方法的原理如下:

1. 将散列算法应用于分布键的值，并生成一个介于零（0）与 4095 之间的编号。
2. 创建数据库分区组时将创建分发映射。每个编号依次以循环方式重复，以填写该分发映射。

3. 将该编号用作分发映射中的索引。分发映射中该位置处的编号是存储该行的数据库分区的编号。

相关概念:

- 第 79 页的『数据库分区组设计』
- 第 78 页的『数据库分区组』
- 第 80 页的『分发映射』
- 『设计顾问程序』（*《性能指南》*）

相关参考:

- 『ALTER TABLE statement』（*SQL Reference, Volume 2*）

表并置

您可能会发现，作为对特定查询的响应，两个或多个表频繁地提供数据。在此情况下，您会希望这样的表中的相关数据的位置尽可能地靠近。在数据库被物理地划分为两个或多个数据库分区的环境中，必须有一种方法可将划分的表的相关碎片尽可能地靠近。完成此过程的功能称为*表并置*。

当表存储在同一个数据库分区组中且它们的分布键兼容时，这些表就是并置的。将两个表置于同一个数据库分区组中以确保存在公共的分发映射。这些表可能位于不同的表空间，但是这些表空间必须与相同数据库分区组相关联。每个分布键中对应列的数据类型必须是*分区兼容*的。

当访问用于连接或子查询的多个表时，DB2 数据库 Linux 版、UNIX 版和 Windows 版能够识别要连接的数据是否位于相同数据库分区上。当发生这种情况时，DB2 可以在存储该数据的数据库分区上执行连接或子查询，而不必在数据库分区之间移动数据。此功能的优点是可以显著改善性能。

相关概念:

- 第 79 页的『数据库分区组设计』
- 第 78 页的『数据库分区组』
- 第 83 页的『数据库分区的兼容性』
- 第 81 页的『分布键』

数据库分区的兼容性

对分布键的对应列的基本数据类型进行比较，并可将它们声明为*分区兼容*。分区兼容的数据类型具有如下属性：具有相同值但不同类型的两个变量会按相同的分区算法映射至同一个编号。

分区兼容性具有下列特征:

- 基本数据类型与另一个相同的基本数据类型兼容。
- 内部格式用于 DATE、TIME 和 TIMESTAMP 数据类型。它们彼此都不兼容，且都不与 CHAR 兼容。
- 分区兼容性不受带有 NOT NULL 或 FOR BIT DATA 定义的列的影响。

- 对兼容数据类型的 NULL 值的处理是完全相同的；对不兼容数据类型的 NULL 值的处理可能不相同。
- 用户定义的类型的基本数据类型用于分析分区兼容性。
- 对分布键中值相同的小数的处理是完全相同的，即使它们的小数位和精度不同也是如此。
- 字符串中（CHAR、VARCHAR GRAPHIC 或 VARGRAPHIC）的尾部空格会被散列算法忽略。
- BIGINT、SMALLINT 和 INTEGER 是兼容的数据类型。
- REAL 和 FLOAT 是兼容的数据类型。
- 不同长度的 CHAR 和 VARCHAR 是兼容的数据类型。
- GRAPHIC 和 VARGRAPHIC 是兼容的数据类型。
- 分区兼容性不适用于 LONG VARCHAR、LONG VARGRAPHIC、CLOB、DBCLOB 和 BLOB 数据类型，因为不支持它们作为分布键。

相关概念:

- 第 79 页的『数据库分区组设计』
- 第 78 页的『数据库分区组』
- 第 81 页的『分布键』

数据分区

数据分区是表的一部分行，这些行不与其他部分的行存储在一起，并且按照 CREATE TABLE 语句的 PARTITION BY 子句中提供的规范分组。如果一个表是使用 PARTITION BY 子句创建的，则该表是分区表。

分区表使用了数据组织方案，即，表数据根据该表中一个或多个表分区键列中的值分布到多个存储对象（称为数据分区或范围）中。根据 CREATE TABLE 语句的 PARTITION BY 子句中指定的内容，给定表的数据被划分到多个存储对象中。这些存储对象可以在不同的表空间中，也可以在相同表空间中。所有指定的表空间在下列方面必须相同：页大小、扩展数据块大小、存储机制（DMS 和 SMS）和类型（常规或大型），并且所有表空间必须位于相同数据库分区组中。

分区表能够简化表数据的转入和转出，并且，与普通的表相比，分区表包含的数据可以多得多。分区表最多可以有 32767 个数据分区。可以对分区表添加数据分区、将数据分区与分区表相连以及断开数据分区与分区表的连接，并且，可以将一个表的多个数据分区范围存储在一个表空间中。

可以自动生成对每个数据分区指定的范围，也可以在创建表时手工生成这些范围。

数据分区在整个 DB2 库中以多种方式被引用。下表列示了最常见的引用方式:

- DATAPARTITIONNAME 是创建给定的表时对数据分区指定的永久名称。此列值存储在 SYSCAT.DATAPARTITIONS 目录视图中。连接或拆离操作不保留此名称。
- DATAPARTITIONID 是创建给定的表时对数据分区指定的永久标识。该标识用于唯一地标识给定表中的特定数据分区。连接或拆离操作不保留此标识。此值由系统生成，它会出现在各个实用程序的输出中。

- SEQNO 表示特定数据分区范围相对于表中其他数据分区范围的顺序，其中，拆离数据分区排列在所有可视数据分区和连接数据分区后面。

相关概念:

- 第 96 页的『DB2 和 Informix 数据库中的数据组织方案』
- 『分区表的优化策略』（*《性能指南》*）
- 第 95 页的『分区表』
- 第 85 页的『表分区功能』

相关任务:

- 『将数据分区添加至分区表』（*《管理指南：实施》*）
- 『改变分区表』（*《管理指南：实施》*）
- 『创建分区表』（*《管理指南：实施》*）
- 『删除数据分区』（*《管理指南：实施》*）
- 『将现有表和视图迁移至分区表的方法』（*《管理指南：实施》*）
- 『连接数据分区』（*《管理指南：实施》*）
- 『拆离数据分区』（*《管理指南：实施》*）
- 『在分区表中旋转数据』（*《管理指南：实施》*）
- 『定义分区表上的范围的方法』（*《管理指南：实施》*）

相关参考:

- 『转入和转出分区表数据的示例』（*《管理指南：实施》*）
- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）
- 『改变带有连接数据分区或拆离数据分区的分区表时的准则和限制』（*《管理指南：实施》*）

表分区功能

表分区功能是一种数据组织方案，即，表数据根据一个或多个表列中的值分布到多个存储对象（称为数据分区或范围）中。每个数据分区都是单独存储的。这些存储对象可以在不同的表空间中，也可以在相同表空间中。

存储对象的行为与单个表的行为很相似，通过使用 ALTER TABLE ...ATTACH 将现有表合并到分区表中，就很容易完成快速转入。同样，使用 ALTER TABLE ...DETACH 语句很容易完成转出。进行查询处理时也可以将数据分隔开，以避免扫描无关数据，从而使大量数据仓库样式查询具有更好的查询性能。

按照 CREATE TABLE 语句的 PARTITION BY 子句中指定那样将表数据分区。此定义中使用的列被称为表分区键列。

这种组织方案可以单独使用，也可以与其他组织方案结合使用。通过组合使用 CREATE TABLE 语句的 DISTRIBUTE BY 和 PARTITION BY 子句，可以在跨多个表空间的数据库分区之间分布数据。DB2 组织方案包括:

- DISTRIBUTE BY HASH
- PARTITION BY RANGE
- ORGANIZE BY DIMENSIONS

表分区功能可用于 DB2 企业服务器版版本 9.1 的 Linux 版、UNIX 版和 Windows 版。

表分区的优势:

如果下列任何一种情况适用于您和您的机构，您就应该考虑使用表分区的巨大优势:

- 您具有能够更方便地转入和转出表数据的数据仓库
- 数据仓库中包含大型的表
- 您正在考虑从 DB2 V9.1 的先前发行版或者从具有竞争力的数据库产品迁移到 DB2 V9.1 数据库
- 您需要更有效率地使用“分层存储管理”（HSM）解决方案

表分区功能简化了表数据转入和转出以及管理工作，并且提高了索引布置灵活性和查询处理效率。

高效率的转入和转出

表分区功能提高了表数据的转入和转出效率。可以通过使用 ALTER TABLE 语句的 ATTACH PARTITION 和 DETACH PARTITION 子句来实现此目标。通过转入分区表数据，可以方便地将新范围作为附加数据分区合并到分区表中。通过转出分区表数据，可以方便地从分区表中分离出某些范围的数据，以便随后进行清除或归档。

更方便地管理大型表

由于可以对各个数据分区执行管理任务，所以表级别的管理工作更为灵活。这些任务包括：拆离和重新连接数据分区、备份和复原各个数据分区以及重组各个索引。可以通过将耗时的维护操作分拆成一系列较小的操作来将其缩短。例如，当数据分区放在不同的表空间中时，可以逐个数据分区地执行备份操作。这样，就有可能每次备份分区表的一个数据分区。

灵活的索引布置

现在，可以将索引放置在不同的表空间中，以允许对索引布置进行更详细的控制。以下是这种新设计的一些优点:

- 提高了删除索引和联机索引创建的性能
- 能够针对每个表索引之间的任何表空间特征使用不同的值（例如，为了确保更好的空间利用率，对每个索引使用不同的页大小可能更合适）。
- 减少 IO 争用并提供对表索引数据更有效的并发访问。
- 删除各个索引时，空间将立即可供系统使用，而无需进行索引重组。
- 如果您选择执行索引重组，可以重组单个索引。

DMS 和 SMS 表空间都支持在不同于表的另一个位置使用索引。

提高了商业智能样式查询的性能

查询处理功能已增强为根据查询谓词自动除去数据分区。此功能称为“数据分区消除”，它可以使许多决策支持查询受益。

以下示例创建一个表 *customer*，其中 l_shipdate >= ’01/01/2006’ 且 l_shipdate <= ’03/31/2006’ 的行存储在表空间 ts1 中，l_shipdate >= ’04/01/2006’ 且 l_shipdate <= ’06/30/2006’ 的行存储在表空间 ts2 中，依此类推。

```
CREATE TABLE customer (l_shipdate, l_name CHAR(30))
IN ts1, ts2, ts3, ts4, ts5
PARTITION BY RANGE(l_shipdate) (STARTING FROM ('01/01/2006')
ENDING AT ('12/31/2006') EVERY (3 MONTHS))
```

相关概念:

- 第 90 页的『数据组织方案』
- 第 84 页的『数据分区』
- 『分区具体化查询表行为』(*《管理指南: 实施》*)
- 第 95 页的『分区表』
- 『了解分区表上的索引行为』(*《性能指南》*)
- 『分区表的优化策略』(*《性能指南》*)
- 『了解分区表上的集群索引行为』(*《性能指南》*)

相关任务:

- 『改变分区表』(*《管理指南: 实施》*)
- 『将数据分区添加至分区表』(*《管理指南: 实施》*)
- 『定义分区表上的范围的方法』(*《管理指南: 实施》*)
- 『将现有表和视图迁移至分区表的方法』(*《管理指南: 实施》*)
- 『连接数据分区』(*《管理指南: 实施》*)
- 『创建分区表』(*《管理指南: 实施》*)
- 『拆离数据分区』(*《管理指南: 实施》*)
- 『删除数据分区』(*《管理指南: 实施》*)
- 『在分区表中旋转数据』(*《管理指南: 实施》*)

相关参考:

- 『转入和转出分区表数据的示例』(*《管理指南: 实施》*)
- 『CREATE TABLE statement』(*SQL Reference, Volume 2*)
- 『Command Line Processor (CLP) samples』(*样本主题*)

表分区键

表分区键是一个或多个表列的有序集合。表分区键列中的值用来确定每个表行所属的数据分区。

要对表定义表分区键，请使用指定了 PARTITION BY 子句的 CREATE TABLE 语句。

选择有效的表分区键列对于充分利用表分区功能的优点来说十分关键。下列准则可以帮助您为分区表选择最有效的表分区键列:

- 将范围定义成与数据转入大小相匹配。最常见的情况是根据日期或时间列对数据进行分区。
- 将范围详细程度定义为与数据转出相匹配。最常见的情况是使用月份或季度。
- 根据有益于消除分区的列进行分区。

支持的数据类型:

支持将下列数据类型(包括同义词)的列用作表分区键列:

SMALLINT	INTEGER
INT	BIGINT
FLOAT	REAL

DOUBLE	DECIMAL
DEC	NUMERIC
NUM	CHARACTER
CHAR	VARCHAR
DATE	TIME
GRAPHIC	VARGRAPHIC
CHARACTER VARYING	TIMESTAMP
CHAR VARYING	CHARACTER FOR BIT DATA
CHAR FOR BIT DATA	VARCHAR FOR BIT DATA
CHARACTER VARYING FOR BIT DATA	CHAR VARYING FOR BIT DATA
用户定义的类型（单值）	

不支持的数据类型:

分区表可以包含下列数据类型，但不支持将它们用作表分区键列:

- 用户定义的类型（结构化）
- LONG VARCHAR
- LONG VARCHAR FOR BIT DATA
- BLOB
- BINARY LARGE OBJECT
- CLOB
- CHARACTER LARGE OBJECT
- DBCLOB
- LONG VARGRAPHIC
- REF
- C 变长字符串
- Pascal 变长字符串

在分区表中，不支持下列数据类型:

- XML
- DATALINK

如果您选择使用 CREATE TABLE 语句的 EVERY 子句来自动生成数据分区，则只能将一列用作表分区键。如果您选择通过在 CREATE TABLE 语句的 PARTITION BY 子句中指定每个范围来手工生成数据分区，则可以将多个列用作表分区键，如以下示例所示:

```
CREATE TABLE sales (year INT, month INT)
 IN tbsp1, tbsp2, tbsp3, tbsp4, tbsp5, tbsp6, tbsp7, tbsp8
 PARTITION BY RANGE(year, month)
 (STARTING FROM (2001, 1) ENDING (2001,3) IN tbsp1,
  ENDING (2001,6) IN tbsp2, ENDING (2001,9)
  IN tbsp3, ENDING (2001,12) IN tbsp4,
  ENDING (2002,3) IN tbsp5, ENDING (2002,6)
  IN tbsp6, ENDING (2002,9) IN tbsp7,
  ENDING (2002,12) IN tbsp8)
```

这将生成 8 个数据分区，即 2001 年和 2002 年的每个季度有一个数据分区。

注:

1. 当将多个列用作表分区键时，将把这些列视为组合键（类似于索引中的组合键），其中，后面的列依赖于前面的列。指定的每个起始值或结束值（所有列一起）不能超出 512 个字符。此限制与 SYSCAT.DATAPARTITIONS 目录视图中的 LOWVALUE 和 HIGHVALUE 列大小对应。如果指定超出 512 个字符的起始值或结束值，就会导致错误 SQL0636N，原因码为 9。
2. 表分区是多列的，而不是多维的。在表分区中，使用的所有列都包含在单个维中。

生成列:

可以将生成列用作表分区键。此示例创建包含 12 个数据分区的表，即每个月一个数据分区。对于任何年份，一月份的所有行都将被放到第一个数据分区中，二月份的行将被放到第二个数据分区中，依此类推。

示例 1

```
CREATE TABLE monthly_sales (sales_date date,
 sales_month int GENERATED ALWAYS AS (month(sales_date)))
     PARTITION BY RANGE (sales_month)
     (STARTING FROM 1 ENDING AT 12 EVERY 1);
```

注:

1. 对于表分区键中使用的生成列，不能改变或删除其表达式。不允许对表分区键中使用的列添加生成列表达式。对于表分区键中使用的列，如果尝试添加、删除或改变该列的生成列表达式，就会导致错误（SQL0270N，原因码为 52）。
2. 如果生成列不是单调的，或者优化器无法检测出该列是否是单调的，就不会对范围谓词使用数据分区消除功能。如果存在非单调表达式，则只能对等价或 IN 谓词执行数据分区消除功能。有关单调性的详细讨论和示例，请参阅多维集群（MDC）表创建、布置和使用。

相关概念:

- 第 85 页的『表分区功能』
- 第 84 页的『数据分区』
- 『分区表的优化策略』（*《性能指南》*）
- 第 95 页的『分区表』

相关任务:

- 『定义分区表上的范围的方法』（*《管理指南：实施》*）
- 『将数据分区添加至分区表』（*《管理指南：实施》*）
- 『创建分区表』（*《管理指南：实施》*）
- 『删除数据分区』（*《管理指南：实施》*）
- 『将现有表和视图迁移至分区表的方法』（*《管理指南：实施》*）
- 『连接数据分区』（*《管理指南：实施》*）
- 『拆离数据分区』（*《管理指南：实施》*）
- 『在分区表中旋转数据』（*《管理指南：实施》*）

相关参考:

- 『SYSCAT.DATAPARTITIONS catalog view』(*SQL Reference, Volume 1*)
- 『转入和转出分区表数据的示例』(《*管理指南: 实施*》)
- 『CREATE TABLE statement』(*SQL Reference, Volume 2*)
- 『DESCRIBE command』(*Command Reference*)
- 『DESCRIBE statement』(*SQL Reference, Volume 2*)

数据组织方案

在引入表分区之后，DB2 数据库提供了一种三级数据组织方案。CREATE TABLE 语句的每个子句都包含一种算法，指示应如何组织数据。下列三个子句演示了可以任意组合在一起使用的数据组织级别:

- DISTRIBUTE BY 用于将数据均匀地分布在数据库分区上（以启用查询内并行性并平衡每个数据库分区上的负载）（数据库分区）
- PARTITION BY 用于对同一个数据分区中具有类似单维值的行进行分组（表分区）
- ORGANIZE BY 用于对同一表扩展数据块中在多个维上具有类似值的行进行分组（多维集群）

此语法允许子句之间保持一致并允许进一步的数据组织算法。每个子句可以单独地使用，也可与其他子句结合使用。通过组合使用 CREATE TABLE 语句的 DISTRIBUTE BY 和 PARTITION BY 子句，可以在跨多个表空间的数据库分区之间分布数据。此方法允许类似于 Informix® Dynamic Server 和 Informix Extended Parallel Server hybrid 功能的行为。

在一个表中，可以组合每个数据组织方案中使用的子句来创建更先进的分区方案。例如，DB2 数据库分区功能部件（DPF）不仅兼容，而且是对表分区的补充。

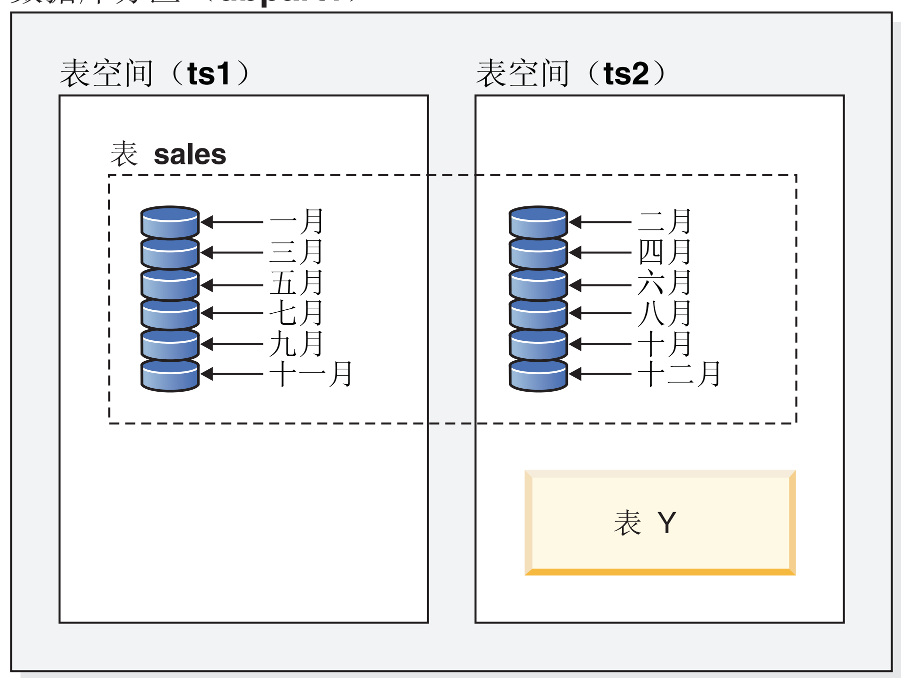
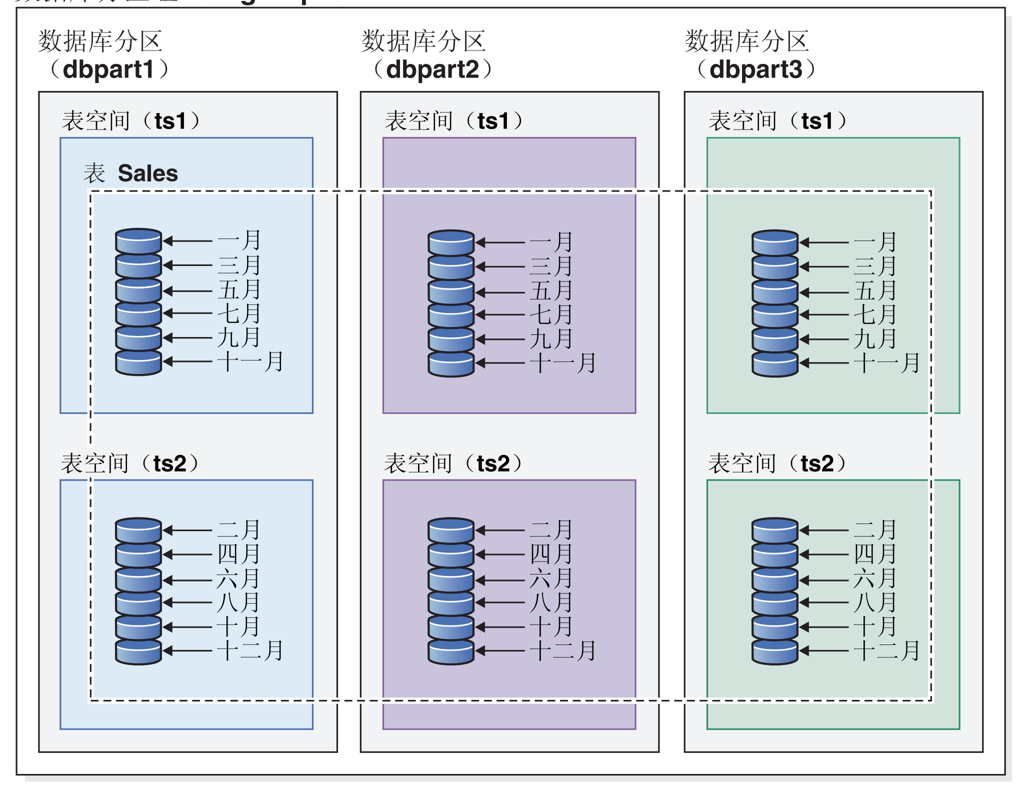




图 24. 演示表分区组织方案，其中表示月销售数据的表划分到多个数据分区中。该表还跨越两个表空间（ts1 和 ts2）。



图注



图 25. 演示数据库分区与表分区相互补充的组织方案。表示月销售数据的表划分到多个数据分区中并且跨两个表空间（ts1 和 ts2），而这两个表空间又分布在数据库分区组（dbgroup1）的多个数据库分区（dbpart1、dbpart2 和 dbpart3）中。

多维集群（MDC）与表分区之间的显著区别是多维和单维。MDC 适合于立方体（即，具有多维的表），而当有一个维是数据库设计的中心时（例如，DATE 列），表分区就能很好地起作用。当同时符合这两个条件时，MDC 和表分区互补。这在第 93 页的图 26 中进行了演示。

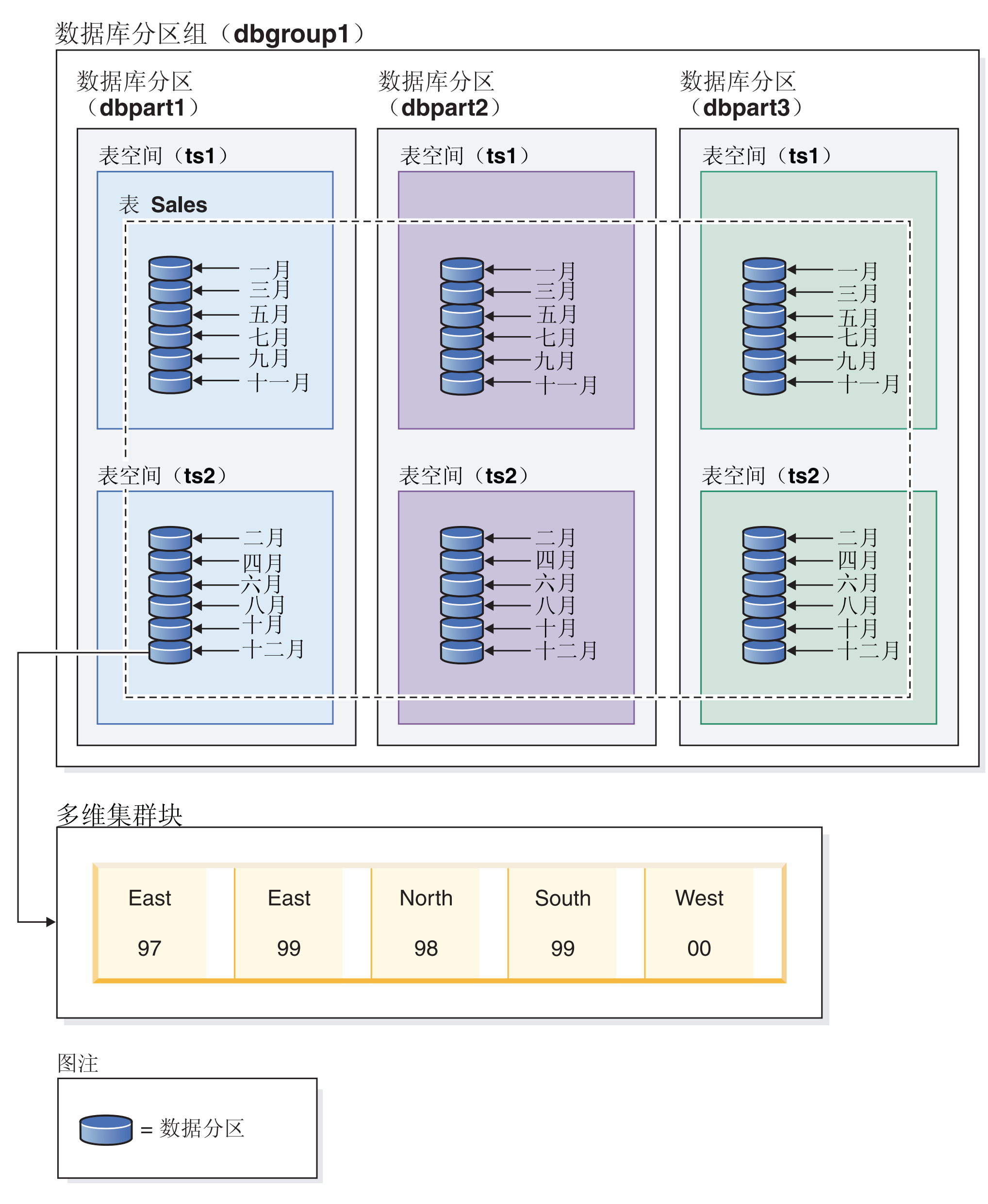


图 26. 数据库分区、表分区和多维组织方案的表示法，其中 SALES 表中的数据不仅分布在多个数据库分区中、划分到表空间 ts1 和 ts2 上，而且还对 date 和 region 维上具有类似值的行进行分组。

还有一种数据组织方案，它不能与上面列示的任何方案一起使用。此方案是 ORGANIZE BY KEY SEQUENCE。它用于将每个记录插入到在创建表（范围集群的表）时为该记录保留的行中。

数据组织术语:

数据库分区（Database partitioning）

一种数据组织方案，即，表数据根据该表中的一个或多个分布键列中的散列值以及使用的数据库分区的分发映射分布到多个数据库分区中。给定表的数据根据 CREATE TABLE 语句的 DISTRIBUTE BY HASH 子句中指定的内容进行分布。

数据库分区（Database partition）
: 数据库分区服务器上的一部分数据库，它由自己的用户数据、索引、配置文件和事务日志组成。数据库分区可以是逻辑或物理的。

表分区（Table partitioning）
: 一种数据组织方案，即，表数据根据该表中一个或多个分区列中的值分布到多个数据分区中。根据 CREATE TABLE 语句的 PARTITION BY RANGE 子句中指定的内容，给定表的数据被划分到多个存储对象中。这些存储对象可以在不同的表空间中。

数据分区（Data partition）
: 表的一部分行，这些行不与其他部分的行存储在一起，并且按照 CREATE TABLE 语句的 PARTITION BY RANGE 子句中指定的内容分组。

多维集群（Multidimensional clustering，MDC）
: 一个表，其数据按 ORGANIZE BY DIMENSIONS 子句中指定的一个或多个维或者集群键以物理方式组织成块。

每种数据组织方案的优势:

了解每种数据组织方案的优势可以帮助您在规划、设计或重新评估数据库系统需求时确定最佳方法。表 21提供了普通客户需求的高级视图，并显示了各种数据组织方案可以如何帮助您满足这些需求。

表 21. 将表分区与数据库分区功能部件配合使用

问题	建议的方案	说明
数据转出	表分区	使用拆离来转出大量数据，并且只出现最少中断。
并行查询执行（查询性能）	数据库分区功能	提供查询并行性以改善查询性能
数据分区消除（查询性能）	表分区	提供数据分区消除以改善查询性能
获得最佳查询性能	两者	一起使用时可以获得最佳查询性能：查询并行性和数据分区消除互补
管理员的工作负载较重	数据库分区功能	为每个数据库分区执行许多任务

表 22. 将表分区与 MDC 表配合使用

问题	建议的方案	说明
转出期间的数据可用性	表分区	使用 DETACH PARTITION 子句来转出大量数据，并且只出现最少中断。
查询性能	两者	MDC 最适合用来查询多个维。表分区通过数据分区消除提高性能。
最少重组	MDC	MDC 维护集群，从而减少进行重组的必要性。

注: 对于 UNION ALL 视图，目前建议使用表分区。

相关概念:

- 第 96 页的『DB2 和 Informix 数据库中的数据组织方案』
- 第 84 页的『数据分区』
- 第 79 页的『数据库分区组设计』
- 第 176 页的『设计多维集群（MDC）表』
- 第 183 页的『多维集群（MDC）表创建、布置和使用』
- 第 159 页的『多维集群表』
- 第 155 页的『范围集群表』
- 第 85 页的『表分区功能』
- 『数据库分区组对查询优化的影响』（*《性能指南》*）
- 『分区表的优化策略』（*《性能指南》*）
- 『范围集群表示例』（*《管理指南: 实施》*）
- 『有关使用范围集群表的准则』（*《管理指南: 实施》*）
- 『Database partitioning across multiple database partitions』（*SQL Reference, Volume 1*）

相关任务:

- 『创建分区表』（*《管理指南: 实施》*）
- 『在分区数据库环境中创建表』（*《管理指南: 实施》*）
- 『在多个表空间中创建表』（*《管理指南: 实施》*）
- 『连接数据分区』（*《管理指南: 实施》*）
- 『拆离数据分区』（*《管理指南: 实施》*）
- 『在分区表中旋转数据』（*《管理指南: 实施》*）

相关参考:

- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）

分区表

分区表使用了数据组织方案，即，表数据根据该表中一个或多个表分区键列中的值分布到多个存储对象（称为数据分区或范围）中。根据 CREATE TABLE 语句的 PARTITION BY 子句中指定的内容，给定表的数据被划分到多个存储对象中。这些存储对象可以在不同的表空间中，也可以在相同表空间中。表分区功能可用于 DB2 企业服务器版版本 9.1 的 Linux 版、UNIX 版和 Windows 版。

表分区功能简化了表数据转入和转出以及管理工作，并且提高了索引布置灵活性和查询处理效率。

不支持分区的分层表或临时表、范围集群表和分区视图。

不支持在分区表中使用以下列类型:

- XML
- DATALINK

相关概念:

- 第 85 页的『表分区功能』
- 第 87 页的『表分区键』
- 第 84 页的『数据分区』
- 第 96 页的『DB2 和 Informix 数据库中的数据组织方案』

相关任务:

- 『将数据分区添加至分区表』（*《管理指南: 实施》*）
- 『改变分区表』（*《管理指南: 实施》*）
- 『改变表』（*《管理指南: 实施》*）
- 『创建分区表』（*《管理指南: 实施》*）
- 『删除数据分区』（*《管理指南: 实施》*）
- 『连接数据分区』（*《管理指南: 实施》*）
- 『拆离数据分区』（*《管理指南: 实施》*）
- 『在分区表中旋转数据』（*《管理指南: 实施》*）
- 『定义分区表上的范围的方法』（*《管理指南: 实施》*）

相关参考:

- 『转入和转出分区表数据的示例』（*《管理指南: 实施》*）
- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）
- 『Command Line Processor (CLP) samples』（*样本主题*）

DB2 和 Informix 数据库中的数据组织方案

表分区功能是一种数据组织方案，即，表数据根据一个或多个表列中的值分布到多个存储对象（称为数据分区或范围）中。每个数据分区都是单独存储的。这些存储对象可以在不同的表空间中，也可以在相同表空间中。按照 CREATE TABLE 语句的 PARTITION BY 子句中指定那样将表数据分区。此定义中使用的列被称为表分区键列。DB2 表分区功能与 Informix Dynamic Server 和 Informix Extended Parallel Server 提供的数据分段组织方法相对应。

Informix 方法:

Informix 支持若干种数据组织方案，在 Informix 产品中，这些方案称为*分段*。其中一种较常使用的分段类型是 FRAGMENT BY EXPRESSION。这种类型的分段的工作方式与 CASE 语句非常相似，其中有一个与表的每个分段相关联的表达式。检查这些表达式以便确定行的放置位置。

Informix 与 DB2 数据库系统的比较:

DB2 数据库提供了丰富的补充功能，这些功能与 Informix 数据组织方案直接对应，这使客户能够相对容易地从 Informix 语法转换为 DB2 语法。DB2 数据库管理器将生成列与 CREATE TABLE 语句的 PARTITION BY RANGE 子句配合使用，以处理复杂的 Informix 方案。第 97 页的表 23 对 Informix 和 DB2 数据库产品中使用的数据组织方案作了比较。

表 23. 所有 Informix 与 DB2 数据组织方案的映射

数据组织方案	Informix 语法	DB2 版本 9.1 语法
• Informix: 基于表达式 • DB2: 表分区	FRAGMENT BY EXPRESSION	PARTITION BY RANGE
• Informix: 循环法 • DB2: 缺省	FRAGMENT BY ROUND ROBIN	没有语法：DB2 数据库管理器自动在容器之间分布数据
• Informix: 范围分布 • DB2: 表分区	FRAGMENT BY RANGE	PARTITION BY RANGE
• Informix: 系统定义的散列 • DB2: 数据库分区	FRAGMENT BY HASH	DISTRIBUTE BY HASH
• Informix: HYBRID • DB2: 在进行表分区的情况下进行数据库分区	FRAGMENT BY HYBRID	DISTRIBUTE BY HASH 和 PARTITION BY RANGE
• Informix: 不适用 • DB2: 多维集群	不适用	ORGANIZE BY DIMENSION

示例:

下列示例详细说明了在 DB2 中如何实现与任何使用表达式的 Informix 分段方案等同的结果。

示例 1: 下面这个基本的 Create Table 语句显示了 Informix 分段以及等同的 DB2 数据库系统表分区语法:

Informix 语法:

```
CREATE TABLE demo(a INT) FRAGMENT BY EXPRESSION
 a = 1 IN db1,
 a = 2 IN db2,
 a = 3 IN db3;
```

DB2 语法:

```
CREATE TABLE demo(a INT) PARTITION BY RANGE(a)
 (STARTING(1) IN db1,
 STARTING(2) IN db2,
 STARTING(3) ENDING(3) IN db3);
```

Informix XPS 支持称为 hybrid 的两层分段方案，在此方案中，使用一个表达式来在协作服务器之间分布数据，并使用第二个表达式来在协作服务器内分布数据。这使所有协作服务器都能够参与查询（即，在所有协作服务器上都有数据），并使查询能够利用数据分区消除功能的优势。

通过结合使用 CREATE TABLE 语句的 DISTRIBUTE BY 和 PARTITION BY 子句，DB2 数据库系统实现了与 Informix hybrid 等同的组织方案。

示例 2: 以下示例显示了组合子句的语法:

Informix 语法

```
CREATE TABLE demo(a INT, b INT) FRAGMENT BY HYBRID HASH(a)
 EXPRESSION b = 1 IN dbsl1,
   b = 2 IN dbsl2;
```

DB2 语法

```
CREATE TABLE demo(a INT, b INT) IN dbsl1, dbsl2
 DISTRIBUTE BY HASH(a),
 PARTITION BY RANGE(b) (STARTING 1 ENDING 2 EVERY 1);
```

此外，可以使用多维集群来进一步组织数据:

```
CREATE TABLE demo(a INT, b INT, c INT) IN dbsl1, dbsl2
 DISTRIBUTE BY HASH(a),
 PARTITION BY RANGE(b) (STARTING 1 ENDING 2 EVERY 1)
 ORGANIZE BY DIMENSIONS(c);
```

这样，列 **a** 值相同的所有行都在同一个数据库分区中。所有列 **b** 值相同的行都在同一个表空间中。对于具有给定的 **a** 和 **b** 值的行，会将再具有相同 **c** 值的所有行集中到一个磁盘上。此方法非常适合于 OLAP 类型的下寻操作，这是因为，仅需扫描单个数据库分区上单个表空间中的一个或数个扩展数据块就可以满足此类查询。

应用表分区以解决常见的应用程序问题:

下列各节讨论如何应用各种 DB2 表分区功能以解决常见的应用程序问题。每一节都特别侧重于采取最佳措施来将各种 Informix 分段方案映射到等同的 DB2 表分区方案。

创建简单数据分区范围时的注意事项:

其中一种最常见的表分区应用是根据日期键对大型事实表进行分区。如果需要创建大小统一的日期范围，请考虑使用自动生成的 CREATE TABLE 语法格式。

示例:

示例 1: 以下示例显示自动生成的语法格式:

```
CREATE TABLE orders
(
 l_orderkey  DECIMAL(10,0) NOT NULL,
 l_partkey  INTEGER,
 l_suppkey  INTEGER,
 l_linenumber  INTEGER,
 l_quantity  DECIMAL(12,2),
 l_extendedprice  DECIMAL(12,2),
 l_discount  DECIMAL(12,2),
 l_tax   DECIMAL(12,2),
 l_returnflag  CHAR(1),
 l_linestatus  CHAR(1),
 l_shipdate  DATE,
 l_commitdate  DATE,
 l_receiptdate  DATE,
 l_shipinstruct  CHAR(25),
 l_shipmode  CHAR(10),
 l_comment  VARCHAR(44))
  PARTITION BY RANGE(l_shipdate)
  (STARTING '1/1/1992' ENDING '12/31/1993' EVERY 1 MONTH);
```

这将创建 24 个范围，即对 1992-1993 的每个月创建一个范围。尝试插入 l_shipdate 超出该范围的行将导致错误。

示例 2: 将上一示例与以下 Informix 语法作比较:

```
create table orders
(
 l_orderkey  decimal(10,0) not null,
 l_partkey  integer,
 l_suppkey  integer,
 l_linenumber integer,
 l_quantity  decimal(12,2),
 l_extendedprice decimal(12,2),
 l_discount  decimal(12,2),
 l_tax   decimal(12,2),
 l_returnflag char(1),
 l_linestatus char(1),
 l_shipdate  date,
 l_commitdate date,
 l_receiptdate date,
 l_shipinstruct char(25),
 l_shipmode  char(10),
 l_comment  varchar(44)
) fragment by expression
l_shipdate < '1992-02-01' in ldbs1,
l_shipdate >= '1992-02-01' and l_shipdate < '1992-03-01' in ldbs2,
l_shipdate >= '1992-03-01' and l_shipdate < '1992-04-01' in ldbs3,
l_shipdate >= '1992-04-01' and l_shipdate < '1992-05-01' in ldbs4,
l_shipdate >= '1992-05-01' and l_shipdate < '1992-06-01' in ldbs5,
l_shipdate >= '1992-06-01' and l_shipdate < '1992-07-01' in ldbs6,
l_shipdate >= '1992-07-01' and l_shipdate < '1992-08-01' in ldbs7,
l_shipdate >= '1992-08-01' and l_shipdate < '1992-09-01' in ldbs8,
l_shipdate >= '1992-09-01' and l_shipdate < '1992-10-01' in ldbs9,
l_shipdate >= '1992-10-01' and l_shipdate < '1992-11-01' in ldbs10,
l_shipdate >= '1992-11-01' and l_shipdate < '1992-12-01' in ldbs11,
l_shipdate >= '1992-12-01' and l_shipdate < '1993-01-01' in ldbs12,
l_shipdate >= '1993-01-01' and l_shipdate < '1993-02-01' in ldbs13,
l_shipdate >= '1993-02-01' and l_shipdate < '1993-03-01' in ldbs14,
l_shipdate >= '1993-03-01' and l_shipdate < '1993-04-01' in ldbs15,
l_shipdate >= '1993-04-01' and l_shipdate < '1993-05-01' in ldbs16,
l_shipdate >= '1993-05-01' and l_shipdate < '1993-06-01' in ldbs17,
l_shipdate >= '1993-06-01' and l_shipdate < '1993-07-01' in ldbs18,
l_shipdate >= '1993-07-01' and l_shipdate < '1993-08-01' in ldbs19,
l_shipdate >= '1993-08-01' and l_shipdate < '1993-09-01' in ldbs20,
l_shipdate >= '1993-09-01' and l_shipdate < '1993-10-01' in ldbs21,
l_shipdate >= '1993-10-01' and l_shipdate < '1993-11-01' in ldbs22,
l_shipdate >= '1993-11-01' and l_shipdate < '1993-12-01' in ldbs23,
l_shipdate >= '1993-12-01' and l_shipdate < '1994-01-01' in ldbs24,
l_shipdate >= '1994-01-01' in ldbs25;
```

注意，Informix 语法提供了上下不封顶的范围以捕获预期范围外的日期。通过添加使用 MINVALUE 和 MAXVALUE 的范围，可以将 DB2 语法修改为与 Informix 语法匹配。

示例 3: 以下示例将示例 1 修改为与 Informix 语法匹配:

```
CREATE TABLE orders
(
 l_orderkey  DECIMAL(10,0) NOT NULL,
 l_partkey  INTEGER,
 l_suppkey  INTEGER,
 l_linenumber INTEGER,
 l_quantity  DECIMAL(12,2),
 l_extendedprice DECIMAL(12,2),
 l_discount  DECIMAL(12,2),
 l_tax   DECIMAL(12,2),
 l_returnflag CHAR(1),
 l_linestatus CHAR(1),
 l_shipdate  DATE,
 l_commitdate DATE,
 l_receiptdate DATE,
 l_shipinstruct CHAR(25),
```

```
 l_shipmode  CHAR(10),
 l_comment  VARCHAR(44)
) PARTITION BY RANGE(l_shipdate)
 (STARTING MINVALUE,
  STARTING '1/1/1992' ENDING '12/31/1993' EVERY 1 MONTH,
  ENDING MAXVALUE);
```

这种技术允许将任何日期插入到表中。

使用生成列按表达式进行分区:

虽然 DB2 数据库并不直接支持按表达式进行分区，但支持按生成列进行分区，因此有可能获得相同的结果。

在决定是否使用此方法前，请考虑下列用法准则:

- 生成列是真实的列，它占用物理磁盘空间。使用生成列的表会略微变大。
- 对于在分区表进行分区时所基于的列，不能改变其生成列表达式。尝试执行此操作将产生消息 SQL0190。如果按下一节描述的方式将新数据分区添加到使用生成列的表中，通常要求改变定义生成列的表达式。目前，不支持改变定义生成列的表达式。
- 当表使用生成列时，对于何时可以应用数据分区消除是有限制的。

示例:

示例 1: 以下示例使用 Informix 语法，在这种情况下适合使用生成列。在本示例中，进行分区时所基于的列存放了加拿大的省和地域。由于省列表不大可能更改，因此生成列表达式也不大可能更改。

```
CREATE TABLE customer (
 cust_id  INT,
 cust_prov CHAR(2))
 FRAGMENT BY EXPRESSION
  cust_prov = "AB" IN dbspace_ab
  cust_prov = "BC" IN dbspace_bc
  cust_prov = "MB" IN dbspace_mb
  ...
  cust_prov = "YT" IN dbspace_yt
  REMAINDER IN dbspace_remainder;
```

示例 2: 在此示例中，使用生成列对 DB2 表进行分区:

```
CREATE TABLE customer (
 cust_id     INT,
 cust_prov   CHAR(2),
 cust_prov_gen GENERATED ALWAYS AS (CASE
  WHEN cust_prov = 'AB' THEN 1
  WHEN cust_prov = 'BC' THEN 2
  WHEN cust_prov = 'MB' THEN 3
  ...
  WHEN cust_prov = 'YT' THEN 13
  ELSE 14 END))
 IN tbspace_ab, tbspace_bc, tbspace_mb, .... tbspace_remainder
 PARTITION BY RANGE (cust_prov_gen)
  (STARTING 1 ENDING 14 EVERY 1);
```

这里，CASE 语句中的表达式与 FRAGMENT BY EXPRESSION 子句中的相应表达式匹配。CASE 语句将每个原始表达式映射到一个数字，该数字存储在生成列（在本示例中是 cust_prov_gen）中。此列是存储在磁盘上的真实列，因此，表占用的空间量会比 DB2 通过表达式直接支持的分区所必需的空间量略多。本示例使用短语法格式。因此，必须在 CREATE TABLE 语句的 IN 子句中列示用来放置数据分区的表空间。如果使用长语法格式，则每个数据分区都需要不同的 IN 子句。

注: 这种技术可以应用于任何 FRAGMENT BY EXPRESSION 子句。

相关概念:

- 第 84 页的『数据分区』
- 第 39 页的『分区数据库环境』
- 第 95 页的『分区表』
- 第 85 页的『表分区功能』
- 『分区表的优化策略』(*《性能指南》*)
- 『了解分区表上的集群索引行为』(*《性能指南》*)
- 『了解分区表上的索引行为』(*《性能指南》*)
- 『拆离的数据分区的属性』(*《管理指南: 实施》*)
- 『Database partitioning across multiple database partitions』(*SQL Reference, Volume 1*)
- 『Large object behavior in partitioned tables』(*SQL Reference, Volume 1*)
- 『分区具体化查询表行为』(*《管理指南: 实施》*)

相关任务:

- 『将数据分区添加至分区表』(*《管理指南: 实施》*)
- 『改变分区表』(*《管理指南: 实施》*)
- 『创建分区表』(*《管理指南: 实施》*)
- 『删除数据分区』(*《管理指南: 实施》*)
- 『在数据库中启用数据库分区』(*《管理指南: 实施》*)
- 『将现有表和视图迁移至分区表的方法』(*《管理指南: 实施》*)
- 『连接数据分区』(*《管理指南: 实施》*)
- 『拆离数据分区』(*《管理指南: 实施》*)
- 『在分区表中旋转数据』(*《管理指南: 实施》*)
- 『定义分区表上的范围的方法』(*《管理指南: 实施》*)

相关参考:

- 『转入和转出分区表数据的示例』(*《管理指南: 实施》*)
- 『CREATE TABLE statement』(*SQL Reference, Volume 2*)
- 『改变带有连接数据分区或拆离数据分区的分区表时的准则和限制』(*《管理指南: 实施》*)

复制具体化查询表

*具体化查询表*是由查询定义的表，也用于确定表中的数据。具体化查询表可用来改进查询的性能。若 DB2 数据库 Linux 版、UNIX 版和 Windows 版确定查询的一部分可使用具体化查询表来解决，则数据库管理器可以重写该查询以使用具体化查询表。

在分区数据库环境中，可以复制具体化查询表并使用它们来改善查询性能。*复制具体化查询表*基于这样一个表：可能已经在单一分区数据库分区组中创建该表，但是您想在另一个数据库分区组中的所有数据库分区之间复制该表。要创建复制具体化查询表，调用带 REPLICATED 关键字的 CREATE TABLE 语句。

通过使用复制具体化查询表，可将未典型并置的表并置。对于大事实表和小维表的连接，复制具体化查询表特别有用。要将所需的额外存储器以及必须更新每个副本所带来的影响降至最小，要复制的表应较小，且更新不频繁。

注: 还应考虑复制那些不常更新的更大的表：复制的单次成本可通过并置获得的性能效益来抵消。

通过在定义复制表所用的 subselect 子句中指定适当的谓词，可以复制选择的列和／或选择的行。

相关概念:

- 第 79 页的『数据库分区组设计』
- 『设计顾问程序』（*《性能指南》*）

相关任务:

- 『创建具体化查询表』（*《管理指南：实施》*）

相关参考:

- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）

表空间设计

表空间是一种存储结构，它包含表、索引、大对象和长型数据。表空间位于数据库分区组中。它们允许将数据库和表数据的位置直接指定到容器上。（容器可以是目录名、设备名或文件名。）这可以提供改善的性能和更灵活的配置。

因为表空间位于数据库分区组中，所以选择保存表的表空间定义了如何将该表的数据分布至数据库分区组中的数据库分区。单个表空间可跨多个容器。在同一个物理磁盘（或驱动器）上创建多个容器（从一个或多个表空间）是可能的。为提高性能，每个容器应使用不同的磁盘。第 103 页的图 27 举例说明了一个数据库内的表和表空间与该数据库相关的容器之间的关系。

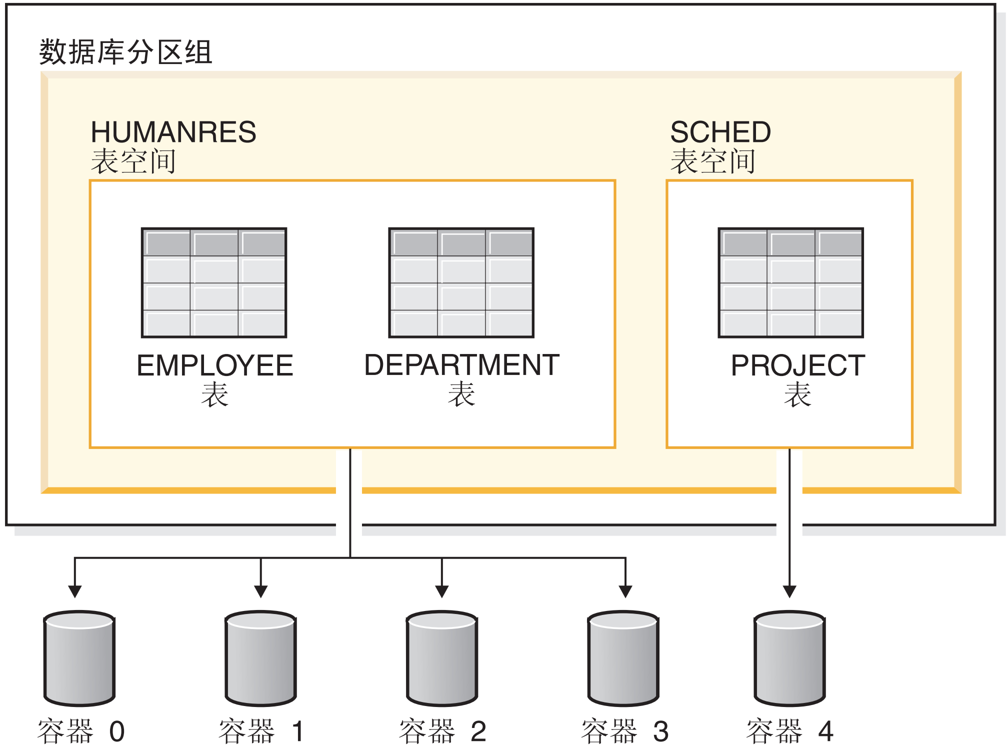


图 27. 数据库中的表空间和表

EMPLOYEE 和 DEPARTMENT 表在 HUMANRES 表空间中，该表空间横跨容器 0、1、2 和 3。PROJECT 表位于容器 4 中的 SCHED 表空间内。此示例显示每个容器存在于单独的磁盘中。

数据库管理器会尝试平衡分布在所有容器中的数据负荷。因此，所有容器都将用于存储数据。数据库管理器在使用另一个容器之前写入一个容器的页数称为*扩展数据块大小*。数据库管理器并非始终从第一个容器开始存储表数据。

第 104 页的图 28 显示具有两个 4KB 页扩展数据块大小的 HUMANRES 表空间，它有四个容器，每个容器有少量已分配的扩展数据块。DEPARTMENT 和 EMPLOYEE 表都有 7 页，且跨所有四个容器。

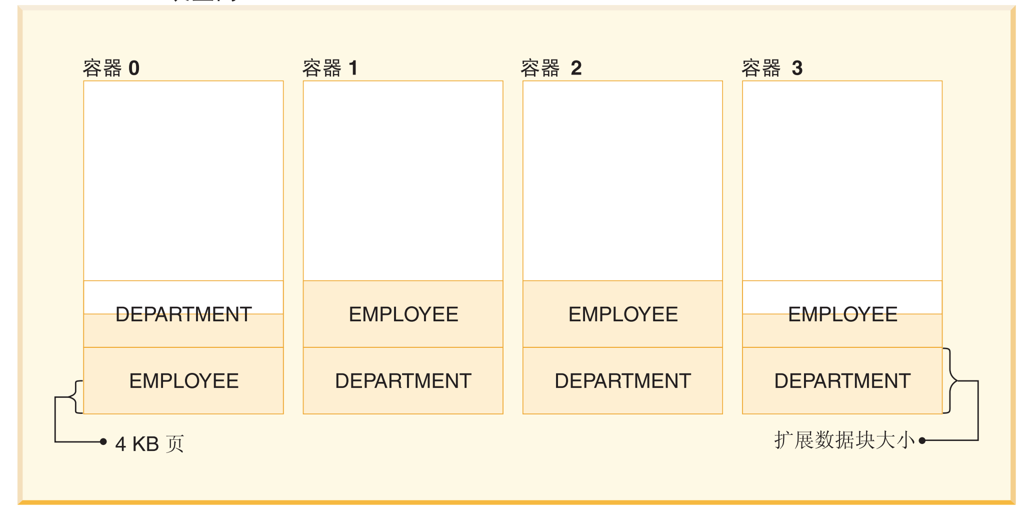


图 28. 表空间中的容器和扩展数据块

一个数据库至少必须包含三个表空间:

- 一个*目录表空间*，它包含该数据库的所有系统目录表。此表空间称为 SYSCATSPACE，它不能被删除。IBMCATGROUP 是此表空间的缺省数据库分区组。
- 一个或多个*用户表空间*，它们包含所有用户定义的表。缺省情况下，会创建一个表空间 USERSPACE1。IBMDEFAULTGROUP 是此表空间的缺省数据库分区组。

  当创建一个表时，应指定表空间名，否则，结果可能不是您所期望的。

  表的页大小或者由行大小确定，或者由列数确定。一行中允许的最大长度取决于创建此表所在的表空间的页大小。可能的页大小的值为 4 KB、8 KB、16 KB 和 32 KB。在版本 9.1 之前，缺省页大小为 4 KB。在版本 9.1 及其后续版本中，缺省页大小可以是其他受支持的值中的一个。缺省页大小是在创建新的数据库时声明的。声明了缺省页大小之后，仍然可以使用具有一种页大小的表空间作为基本表，而使用具有另一种页大小的另一个表空间来存储长型数据或 LOB 数据。（记住，SMS 不支持跨表空间的表，而 DMS 却支持。）若列数或行大小超过表空间页大小的限制，则返回一个错误（SQLSTATE 42997）。
- 一个或多个*临时表空间*，它们包含临时表。临时表空间可以是*系统临时表空间*或*用户临时表空间*。

  系统临时表空间存放数据库管理器在执行诸如排序或连接之类的操作时所需的临时数据。这些类型的操作需要额外的空间来处理结果集。数据库必须有至少一个系统临时表空间；在缺省情况下，创建数据库时会创建一个名为 TEMPSPACE1 的系统临时表空间。IBMTEMPGROUP 是此表空间的缺省数据库分区组。

  用户临时表空间存放使用 DECLARE GLOBAL TEMPORARY TABLE 语句创建的表的临时数据。为了能够定义已声明临时表，至少一个用户临时表空间应该是使用相应 USE 特权创建的。USE 特权是使用 GRANT 语句授予的。用户临时表空间不是在创建数据库时缺省创建的。

  如果数据库使用多个临时表空间，并且需要新的临时对象，则优化器将为此对象选择相应的页大小。然后将把该对象分配到具有相应页大小的临时表空间中。若存在

多个临时表空间具有该页大小，则将以循环方式选择表空间。在大多数情况下，建议对于任何一个页大小，不要使用多个临时表空间。

如果对用大于缺省值的页大小定义的表空间中的表运行查询，则某些查询可能会失败。若没有使用更大页大小定义的临时表空间，则会发生这种情况。可能需要创建具有更大页大小的临时表空间（如果缺省值是 4 KB，则需要创建页大小为 8 KB、16 KB 或 32 KB 的临时表空间）。若没有与用户表空间中的最大页大小相同的临时表空间，任何 DML（数据操作语言）语句都可能失败。

应定义一个 SMS 临时表空间，使它的页大小等于大多数用户表空间所使用的页大小。这对于典型的环境和工作负载应已足够。

在分区数据库环境中，目录节点将包含全部三个缺省表空间，而其他每个数据库分区将只包含 TEMPSPACE1 和 USERSPACE1。

有两种类型的表空间，它们都可以在单个数据库中使用:
- 系统管理的空间，操作系统的文件管理器控制其中的存储空间。
- 数据库管理的空间，数据库管理器控制其中的存储空间。

相关概念:

- 第 151 页的『目录表空间设计』
- 第 131 页的『SMS 和 DMS 表空间的比较』
- 第 110 页的『数据库管理的空间』
- 第 134 页的『扩展数据块大小』
- 第 135 页的『表空间和缓冲池之间的关系』
- 第 136 页的『表空间和数据库分区组之间的关系』
- 第 107 页的『系统管理的空间』
- 第 105 页的『SYSTOOLSPACE 和 SYSTOOLSTMPSPACE 表空间』
- 第 149 页的『临时表空间设计』
- 第 133 页的『表空间设计中的工作负载注意事项』
- 第 132 页的『表空间磁盘 I/O』
- 『Table spaces and other storage structures』（*SQL Reference, Volume 1*）

相关任务:

- 第 152 页的『当数据在 RAID 设备上时优化表空间性能』
- 『创建表空间』（*《管理指南: 实施》*）

相关参考:

- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）
- 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）

SYSTOOLSPACE 和 SYSTOOLSTMPSPACE 表空间

SYSTOOLSPACE 表空间是一个用户数据表空间，DB2 管理工具和一些 SQL 管理例程使用它来存储历史数据和配置信息。下列工具和 SQL 管理例程使用 SYSTOOLSPACE 表空间:
- 设计顾问程序

- “改变表”笔记本
- “配置自动维护”向导
- 存储管理工具
- db2look 命令
- 自动收集统计信息（包括统计信息收集所需的运行状况指示器）
- 自动重组（包括重组所需的运行状况指示器）
- GET_DBSIZE_INFO 存储过程
- ADMIN_COPY_SCHEMA 存储过程
- ADMIN_DROP_SCHEMA 存储过程
- SYSINSTALLOBJECTS 存储过程
- ALTOBJ 存储过程

SYSTOOLSPACE 表空间是第一次使用上述任一工具和例程（DB2LOOK、ALTOBJ、ADMIN_COPY_SCHEMA 和 ADMIN_DROP_SCHEMA 除外）时创建的。

SYSTOOLSTMPSPACE 表空间是一个用户临时表空间，REORGCHK_TB_STATS、REORGCHK_IX_STATS 和 ADMIN_CMD 存储过程使用它来存储临时数据。SYSTOOLSTMPSPACE 表空间是第一次调用其中任一存储过程（ADMIN_CMD 除外）时创建的。

注:

1. 如果 DB2 注册表变量 DB2_WORKLOAD 设置为 SAP，则既不会自动创建 SYSTOOLSPACE，也不会创建 SYSTOOLSTMPSPACE。
2. 缺省情况下，会对所有新数据库启用重组所需和统计信息收集所需的运行状况指示器和运行状况监视器。这两个运行状况指示器由运行状况监视器大约每两个小时评估一次。这表示在新数据库处于活动状态两个小时之后，将自动为它们创建 SYSTOOLSPACE 和 SYSTOOLSTMPSPACE 表空间，否则将显式禁用运行状况监视器或这些运行状况指示器。
3. 缺省情况下，会对所有新数据库启用自动收集统计信息功能。此功能大约每两个小时评估一次。这表示在新数据库处于活动状态两个小时之后，将自动为它们创建 SYSTOOLSPACE 和 SYSTOOLSTMPSPACE 表空间，否则将显式禁用自动收集统计信息功能。

如果不想要任一表空间的缺省定义，则可以手工创建表空间（如果已经自动创建了这些表空间，则删除它们，然后重新创建）。表空间定义可能不同（例如，可以使用 DMS 或 SMS 表空间，或者可以启用或禁用自动存储器），但是必须在 IBMCATGROUP 数据库分区组中创建表空间。如果尝试在任何其他数据库分区组中创建它们，则将会返回错误 SQL1258N。

以下是如何手工创建 SYSTOOLSPACE 和 SYSTOOLSTMPSPACE 表空间的一个示例。此示例使用自动创建表空间时使用的那些定义:

如果数据库使用自动存储器:

```
CREATE TABLESPACE SYSTOOLSPACE IN IBMCATGROUP
  MANAGED BY AUTOMATIC STORAGE
  EXTENTSIZE 4
```

```
CREATE USER TEMPORARY TABLESPACE SYSTOOLSTMPSPACE IN IBMCATGROUP
  MANAGED BY AUTOMATIC STORAGE
  EXTENTSIZE 4
```

如果数据库不使用自动存储器:

```
CREATE TABLESPACE SYSTOOLSPACE IN IBMCATGROUP
  MANAGED BY DATABASE USING ( FILE 'SYSTOOLSPACE' 32 M )
  AUTORESIZE YES
  EXTENTSIZE 4

CREATE USER TEMPORARY TABLESPACE SYSTOOLSTMPSPACE IN IBMCATGROUP
  MANAGED BY SYSTEM USING ( 'SYSTOOLSTMPSPACE' )
  EXTENTSIZE 4
```

缺省情况下，只要数据库不是使用受限访问权创建的，就会授予 PUBLIC 组使用 SYSTOOLSTMPSPACE 的权限。

相关概念:

- 『设计顾问程序』（*《性能指南》*）
- 第 30 页的『自动重组』

相关任务:

- 『使用自动收集统计信息』（*《性能指南》*）
- 『改变表』（*《管理指南：实施》*）

相关参考:

- 『ADMIN_COPY_SCHEMA procedure – Copy a specific schema and its objects』（*Administrative SQL Routines and Views*）
- 『ADMIN_DROP_SCHEMA procedure – Drop a specific schema and its objects』（*Administrative SQL Routines and Views*）
- 『ALTOBJ procedure』（*Administrative SQL Routines and Views*）
- 『db2look - DB2 statistics and DDL extraction tool command』（*Command Reference*）
- 『运行状况指示器总结』（*《系统监视器指南和参考》*）
- 第 137 页的『“存储管理”视图』
- 『GET_DBSIZE_INFO procedure』（*Administrative SQL Routines and Views*）
- 『SYSINSTALLOBJECTS procedure』（*Administrative SQL Routines and Views*）

系统管理的空间

在 SMS（系统管理的空间）表空间中，操作系统的文件系统管理器分配和管理用于存储表的空间。该存储模型通常由存储在文件系统空间中的多个文件组成，这些文件表示表对象。用户决定文件的位置、DB2 数据库 Linux 版、UNIX 版和 Windows 版控制它们的名称，而文件系统负责管理它们。通过控制写入每个文件的数据量，数据库管理器均匀地将数据分布到所有表空间容器中。

每个表至少有一个与它相关的 SMS 物理文件。

表空间中的数据按系统中所有容器上的扩展数据块进行条带分割。*扩展数据块*是对数据库定义的一组连续页。文件扩展名表示该文件中存储的数据的类型。为了在表空间

中的所有容器上平均分布数据，表的起始扩展数据块以循环方式分布在所有容器上。如果数据库中包含许多容量较小的表，则这种扩展数据块分布特别重要。

在 SMS 表空间中，表的空间大小是按需分配的。分配的空间量取决于 *multipage_alloc* 数据库配置参数的设置。如果将此配置参数设置为 YES，则需要空间时将分配完整的扩展数据块（通常包含两页或更多页）。否则，一次分配的空间将为一页。

缺省情况下，启用了多页文件分配功能。multipage_alloc 数据库配置参数值将指示是否已启用多页文件分配功能。

注: 多页文件分配功能不适用于临时表空间。

多页文件分配将只影响一个表的数据和索引部分。这意味着 .LF、.LB 和 .LBA 文件并不会一次扩展一个扩展数据块。

在 SMS 表空间中，当将单个容器中的所有空间都分配给表时，就认为该表空间“已满”，即使其他容器中还有剩余空间。仅当数据库分区中没有任何容器时，才能向该数据库分区中的 SMS 表空间添加容器。

注: SMS 表空间可以利用文件系统预取和高速缓存的功能。

SMS 表空间是使用 CREATE DATABASE 命令上或 CREATE TABLESPACE 语句上的 MANAGED BY SYSTEM 选项定义的。当设计 SMS 表空间时，必须考虑两个关键要素:

- 表空间的容器。

  必须指定要用于表空间的容器数。标识要使用的所有容器是非常重要的，因为您不能在创建了 SMS 表空间之后添加或删除容器。在分区数据库环境中，在将新数据库分区添加至 SMS 表空间的数据库分区组时，可以使用 ALTER TABLESPACE 语句将容器添加至新的数据库分区。

  用于一个 SMS 表空间的每个容器都标识一个绝对或相对目录名。其中每一个目录都可以位于不同的文件系统（物理磁盘）上。表空间的最大大小可以按以下方法估计:

  ```
  容器数 * (操作系统支持的最大文件系统大小)
  ```

  此公式假定有一个唯一的文件系统映射至每个容器，且每个文件系统都具有最大量的可用空间。实际上，情况可能不是这样，表空间最大大小可能小得多。对于数据库对象的大小也有 SQL 限制，它可能影响表空间的最大大小。

  注: 定义容器时必须很小心。若容器上已有文件或目录，将返回一个错误（SQL0298N）。

- 表空间的扩展数据块大小。

  扩展数据块大小只能在创建表空间时指定。因为以后不能更改它，因此为扩展数据块大小选择一个适当的值就很重要。

  创建表空间时，如果不指定扩展数据块大小，数据库管理器将使用缺省扩展数据块大小来创建表空间，该缺省大小由 *dft_extent_sz* 数据库配置参数定义。此配置参数最初是根据创建该数据库时提供的信息设置的。若未在 CREATE DATABASE 命令上指定 *dft_extent_sz* 参数，则会将缺省扩展数据块大小设置为 32。

要为表空间的容器数和扩展数据块大小选择适当的值，必须了解:

- 操作系统对逻辑文件系统的大小施加的限制。

  例如，某些操作系统有 2GB 的限制。因此，若想要一个 64GB 的表对象，则在此类型的系统上将需要至少 32 个容器。

  当创建该表空间时，可以指定位于不同文件系统上的容器，以便增加可以存储在该数据库中的数据量。

- 数据库管理器如何管理与一个表空间相关的数据文件和容器。

  在为该表空间指定的第一个容器中创建第一个表数据文件（SQL00001.DAT），并允许此文件增大至该扩展数据块大小。当它达到此大小之后，数据库管理器将数据写入下一个容器中的 SQL00001.DAT。此过程会继续，直到所有容器都包含 SQL00001.DAT 文件为止，在那时，数据库管理器会返回至第一个容器。此过程（称为*条带分割*）会继续在容器目录中运行，直到一个容器装满为止（SQL0289N）或操作系统中不再有空间可分配为止（磁盘已满错误）。条带分割也用于索引（SQL*nnnnn*.INX）、长型字段（SQL*nnnnn*.LF）、 LOB（SQL*nnnnn*.LB 和 SQL*nnnnn*.LBA）以及 XML（SQL*nnnnn*.XDA）文件。

  注: 只要任何一个容器已满，SMS 表空间就满了。因此，每个容器具有相同容量的可用空间是很重要的。

  为了有助于将数据更加均匀地分布至这些容器中，数据库管理器根据将表标识（以上示例中为 SQL00001.DAT）除以容器数所得的余数来确定首先使用的容器。容器从 0 开始依次编号。

相关概念:

- 第 131 页的『SMS 和 DMS 表空间的比较』
- 第 110 页的『数据库管理的空间』
- 第 102 页的『表空间设计』

相关参考:

- 『db2empfa - Enable multipage file allocation command』（*Command Reference*）
- 『multipage_alloc － 启用的多页文件分配配置参数』（*《性能指南》*）

SMS 表空间

“系统管理的空间”（SMS）表空间用于存储操作系统文件中的数据。表空间中的数据按系统中所有容器上的扩展数据块进行条带分割。*扩展数据块*是对数据库定义的一组连续页。文件扩展名表示该文件中存储的数据的类型。为了在表空间中的所有容器上平均分布数据，表的起始扩展数据块以循环方式分布在所有容器上。如果数据库中包含许多容量较小的表，则这种扩展数据块分布特别重要。

在 SMS 表空间中，表的空间大小是按需分配的。分配的空间量取决于 *multipage_alloc* 数据库配置参数的设置。如果将此配置参数设置为 YES，则需要空间时将分配完整的扩展数据块（通常包含两页或更多页）。否则，一次分配的空间将为一页。缺省情况下，启用了多页文件分配功能。在版本 8.2 之前，该配置参数的缺省设置为 NO，它将导致一次分配一页。可以使用 db2empfa 工具来更改此缺省值，该工具允许启用多页文件分配功能。运行 db2empfa 时，multipage_alloc 数据库配置参数将被设置为 YES。

注: 多页文件分配功能不适用于临时表空间。

多页文件分配将只影响一个表的数据和索引部分。这意味着 .LF、.LB 和 .LBA 文件并不会一次扩展一个扩展数据块。

在 SMS 表空间中，当将单个容器中的所有空间都分配给表时，就认为该表空间“已满”，即使其他容器中还有剩余空间。仅当数据库分区中没有任何容器时，才能向该数据库分区中的 SMS 表空间添加容器。

注: SMS 表空间可以利用文件系统预取和高速缓存的功能。

相关概念:

- 第 131 页的『SMS 和 DMS 表空间的比较』
- 第 102 页的『表空间设计』

相关任务:

- 『将容器添加到数据库分区上的 SMS 表空间』（*《管理指南: 实施》*）

相关参考:

- 『db2empfa - Enable multipage file allocation command』（*Command Reference*）
- 『multipage_alloc － 启用的多页文件分配配置参数』（*《性能指南》*）

数据库管理的空间

在 DMS（数据库管理的空间）表空间中，数据库管理器控制存储空间。存储模型由有限数目的设备或文件组成，这些设备或文件的空间由 DB2 数据库 Linux 版、UNIX 版和 Windows 版来管理。数据库管理员决定使用哪些设备和文件，而 DB2 管理这些设备和文件上的空间。这种表空间实质上实现了一种特殊用途的文件系统，用于最好地满足数据库管理器的需要。

DMS 表空间与 SMS 表空间之间的差异在于，对于 DMS 表空间，空间是在创建表空间时分配的。对于 SMS 表空间，空间是根据需要分配的。包含用户定义的表和数据的 DMS 表空间可以定义为存储任何表数据或索引数据的*常规*表空间或*大型*表空间。

在设计 DMS 表空间和容器时，应该考虑下列事项:

- 数据库管理器使用“拆开”来确保数据均匀地分布在所有容器上。
- 常规表空间的最大大小是 512 GB，每页的大小为 32 KB。大型表空间的最大大小是 16TB。有关其他页大小的常规表空间的最大大小，请参阅 SQL and XQuery limits。
- 与 SMS 表空间不同，组成一个 DMS 表空间的容器不必大小相同；然而，通常不建议这样做，因为会导致在容器间不均匀地进行条带分割，并会降低性能。若任何容器已满，DMS 表空间会使用其他容器中的可用空间。
- 因为空间是预分配的，所以在能够创建表空间之前，该空间必须是可用的。当使用设备容器时，该设备也必须有足够的空间以便存储容器定义。每个设备上只能定义一个容器。为避免浪费空间，设备的大小应该等于容器的大小。例如，若分配给该设备 5000 页，而将设备容器定义为分配 3000 页，则设备上的 2000 页将是不可用的。

- 在缺省情况下，每个容器中都保留一个扩展数据块作为开销。只使用整个扩展数据块，因此为了对空间进行最优管理，可以使用如下公式来帮助您确定当分配容器时要使用的适当大小:

  ```
  extent_size * (n + 1)
  ```

  其中，*extent_size* 是表空间中每个扩展数据块的大小，而 *n* 是您要在该容器中存储的扩展数据块数目。
- DMS 表空间的最小大小是五个扩展数据块。试图创建小于五个扩展数据块的表空间将产生错误（SQL1422N）。
  - 表空间中有三个扩展数据块是保留给开销使用的。
  - 要存储任何用户表数据，至少需要两个扩展数据块。（这些扩展数据块是一个表的规则数据所必需的，但不是任何索引、长型字段或大对象数据所需的，它们需要自己的扩展数据块。）
- 设备容器必须使用带“字符型特殊接口”的逻辑卷，而不是物理卷。
- 在 DMS 表空间中可以使用文件以代替设备。文件与设备之间不存在操作方面的区别；但是，由于存在与文件系统相关的运行时开销，文件效率会更低。在下列情况下，适合使用文件:
  - 设备不直接受支持
  - 设备不可用
  - 不需要最大性能
  - 不想设置设备
- 若工作负载涉及 LOB 或 LONG VARCHAR 数据，则可通过文件系统高速缓存改进性能。

  注: 数据库管理器的缓冲池不缓冲 LOB 和 LONG VARCHAR。
- 某些操作系统允许拥有大小超过 2GB 的物理设备。应该考虑将物理设备划分为多个逻辑设备，以使任何容器都不超过操作系统允许的大小。

注: 与 SMS 表空间相似，DMS 文件容器可以利用文件系统预取和高速缓存。然而，使用原始设备容器的 DMS 表空间却不能。

这一关于页在存储器中的连续放置的一般性陈述有一个例外。当使用 DMS 表空间时，有两个容器选项: 原始设备和文件。当使用文件容器时，数据库管理器在创建表空间时分配整个容器。这种一开始就分配整个表空间的结果是，即使由文件系统执行分配，物理分配也通常（但不保证）是连续的。当使用原始设备容器时，数据库管理器控制整个设备，并总是确保扩展数据块中的页是连续的。

当使用 DMS 表空间时，您应考虑将每个容器与不同的磁盘相关联。这使表空间容量可以更大，并且能够利用并行 I/O 操作。

CREATE TABLESPACE 语句在数据库中创建新的表空间，向表空间分配容器，并在目录中记录表空间定义和属性。当创建表空间时，扩展数据块大小被定义为许多连续页。扩展数据块是表空间中的空间分配单位。只有一个表或对象（例如，索引）能够使用任何一个扩展数据块中的页。将逻辑表空间地址映射中的扩展数据块分配给表空间中创建的所有对象。扩展数据块分配通过“空间映射页”（SMP）进行管理。

逻辑表空间地址映射中的第一个扩展数据块是包含内部控制信息的表空间的头部。第二个扩展数据块是表空间的“空间映射页”（SMP）的第一个扩展数据块。SMP 扩展数

据块以固定间隔方式分布在表空间中。每个 SMP 扩展数据块都是当前 SMP 扩展数据块到下一 SMP 扩展数据块的扩展数据块位映射。位映射用来跟踪正在使用哪些中间扩展数据块。

SMP 后面的一个扩展数据块是表空间的对象表。对象表是一个内部表，它跟踪表空间中存在哪些用户对象，以及它们的第一个“扩展数据块映像页”（EMP）扩展数据块的位置。每个对象都有其自己的 EMP，它提供了指向该对象的每一页的映射，这些映射存储在逻辑表空间地址映射中。图 29 说明如何在逻辑表空间地址映射中分配扩展数据块。

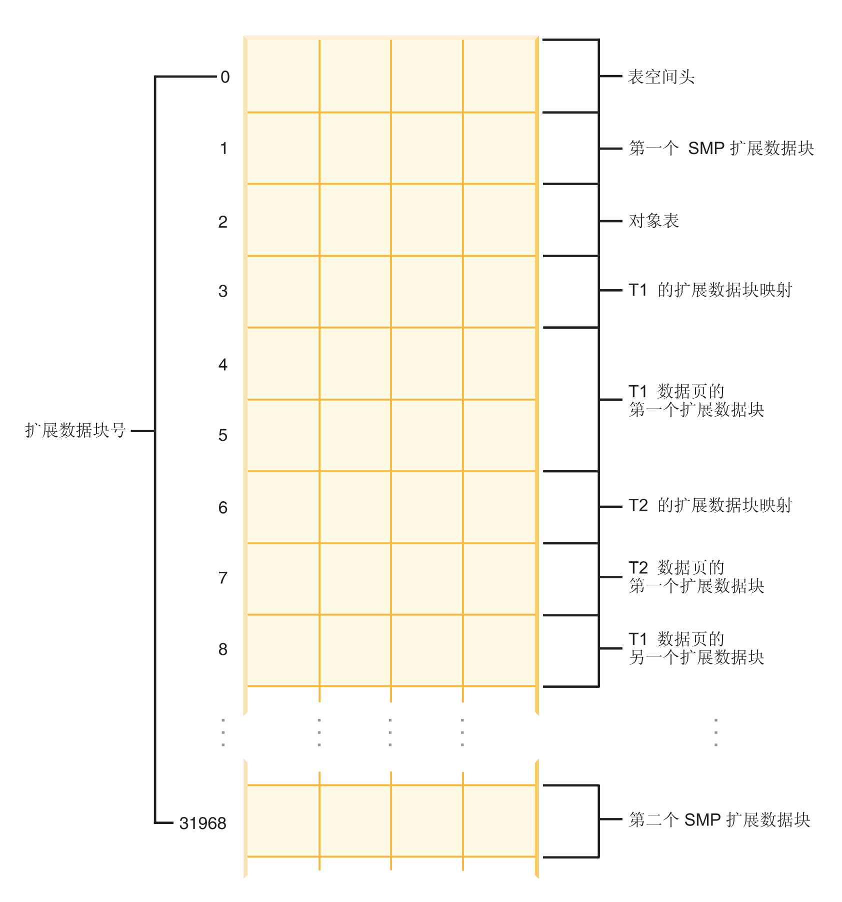


图 29. 逻辑表空间地址映射

相关概念:

- 第 131 页的『SMS 和 DMS 表空间的比较』
- 第 119 页的『在 DMS 表空间中如何添加和扩展容器』
- 第 107 页的『系统管理的空间』
- 第 102 页的『表空间设计』
- 第 115 页的『表空间映射』

DMS 表空间

数据库管理器通过数据库管理的空间（DMS）表空间控制存储空间。当定义 DMS 表空间时，选择设备或文件的列表，使其属于该表空间。那些设备或文件上的空间由 DB2 数据库管理器管理。对于 SMS 表空间和容器，DMS 表空间和数据库管理器使用“按扩展数据块进行条带分割”来确保数据均匀地分布在所有容器上。

DMS 表空间与 SMS 表空间之间的差异在于，对于 DMS 表空间，空间是在创建表空间时分配的，而不是在需要时分配的。

并且，在这两种类型的表空间上，数据的放置也会有所不同。例如，考虑高效表扫描的这样的一个需要：重要的是扩展数据块中的页在物理上是连续的。对于 SMS，操作系统的文件系统决定了每个逻辑文件页的物理放置位置。根据文件系统上其他活动的级别以及用来确定放置位置的算法的不同，这些页可能连续分配，也可能不连续分配。但是，对于 DMS，因为数据库管理器直接与磁盘打交道，所以它可以确保这些页在物理上是连续的。

注: 与 SMS 表空间相似，DMS 文件容器可以利用文件系统预取和高速缓存。然而，DMS 表空间不能。

这一关于页在存储器中的连续放置的一般性陈述有一个例外。当使用 DMS 表空间时，有两个容器选项：原始设备和文件。当使用文件容器时，数据库管理器在创建表空间时分配整个容器。这种一开始就分配整个表空间的结果是，即使由文件系统执行分配，物理分配也通常（但不保证）是连续的。当使用原始设备容器时，数据库管理器控制整个设备，并总是确保扩展数据块中的页是连续的。

与 SMS 表空间不同，组成一个 DMS 表空间的容器的容量不必接近相等。但是，建议让容器的容量相等，或接近相等。并且，如果任何容器已满，则 DMS 表空间可以使用其他容器中的任何可用的可用空间。

当使用 DMS 表空间时，您应考虑将每个容器与不同的磁盘相关联。这使表空间容量可以更大，并且能够利用并行 I/O 操作。

CREATE TABLESPACE 语句在数据库中创建新的表空间，向表空间分配容器，并在目录中记录表空间定义和属性。当创建表空间时，扩展数据块大小被定义为许多连续页。扩展数据块是表空间中的空间分配单位。只有一个表或其他对象（例如，索引）能够使用任何一个扩展数据块中的页。将逻辑表空间地址映射中的扩展数据块分配给表空间中创建的所有对象。扩展数据块分配通过“空间映射页”（SMP）进行管理。

逻辑表空间地址映射中的第一个扩展数据块是包含内部控制信息的表空间的头部。第二个扩展数据块是表空间的“空间映射页”（SMP）的第一个扩展数据块。SMP 扩展数据块以固定间隔方式分布在表空间中。每个 SMP 扩展数据块仅仅是当前 SMP 扩展数据块到下一 SMP 扩展数据块的扩展数据块位映射。位映射用来跟踪正在使用哪些中间扩展数据块。

SMP 后面的一个扩展数据块是表空间的对象表。对象表是一个内部表，它跟踪表空间中存在哪些用户对象，以及它们的第一个“扩展数据块映像页”（EMP）扩展数据块的位置。每个对象都有其自己的 EMP，它提供了指向该对象的每一页的映射，这些映射存储在逻辑表空间地址映射中。

相关概念:

- 第 131 页的『SMS 和 DMS 表空间的比较』
- 第 67 页的『数据库目录和文件』
- 第 110 页的『数据库管理的空间』
- 第 114 页的『DMS 设备注意事项』
- 第 109 页的『SMS 表空间』
- 第 102 页的『表空间设计』

相关任务:

- 『将容器添加至 DMS 表空间』(*《管理指南: 实施》*)

相关参考:

- 『CREATE TABLESPACE statement』(*SQL Reference, Volume 2*)

DMS 设备注意事项

如果对表空间使用“数据库管理存储器”(DMS)设备容器，考虑下列因素以便有效管理:

- **文件系统高速缓存**

  执行文件系统高速缓存，如下所示:

  – 对于 DMS 文件容器(和所有 SMS 容器)，操作系统可能会将页高速缓存在文件系统高速缓存中

  – 对于 DMS 设备容器表空间，操作系统不会将页高速缓存在该文件系统高速缓存中。

  注: 在 Windows 上，注册表变量 DB2NTNOCACHE 指定 DB2 是否用 NOCACHE 选项打开数据库文件。如果 DB2NTNOCACHE=ON，则消除文件系统高速缓存。如果 DB2NTNOCACHE=OFF，则操作系统高速缓存 DB2 文件。这适用于除包含长型字段或 LOB 的文件以外的所有数据。消除系统高速缓存使数据库有更多内存可用，这样可以增加缓冲池或排序堆。

- **数据的缓冲**

  从磁盘读取的表数据通常可在数据库的缓冲池中找到。在某些情况下，在应用程序实际使用一个数据页之前，可能从缓冲池释放该页，特别在其他数据页需要缓冲池空间时，更是如此。对于使用系统管理存储器(SMS)或数据库管理存储器(DMS)文件容器的表空间，以上文件系统高速缓存可以消除另外将需要的 I/O。

  使用数据库管理存储器(DMS)设备容器的表空间不使用文件系统或其高速缓存。因此，您可以增大数据库缓冲池的大小，减小文件系统高速缓存的大小，以修正这样一个事实，即，使用设备容器的 DMS 表空间并未执行双缓冲区。

  如果系统级别的监视工具显示: 与等效的 SMS 表空间相比，使用设备容器的 DMS 表空间的 I/O 更高，这种差别可能是由于双缓冲区所导致的。

- **使用 LOB 或 LONG 数据**

当一个应用程序检索 LOB 或 LONG 数据时，数据库管理器不在它的缓冲区中高速缓存该数据，每次应用程序需要其中一个页时，数据库管理器必须从磁盘检索它。但是，如果 LOB 或 LONG 数据存储在 SMS 或 DMS 文件容器中，文件系统高速缓存可提供缓冲，因此也就改善了性能。

因为系统目录包含一些 LOB 列，所以应该将它们保存在 SMS 表空间或 DMS 文件表空间中。

相关概念:

- 第 113 页的『DMS 表空间』
- 第 109 页的『SMS 表空间』
- 第 67 页的『数据库目录和文件』

表空间映射

表空间映射是 DB2 V9.1 的 DMS 表空间的内部表示法，它描述表空间中页位置的逻辑至物理转换。以下信息描述表空间映射为何有用，以及表空间映射中的信息从何而来。

在 DB2 数据库 Linux 版、UNIX 版和 Windows 版数据库中，DMS 表空间中的页按照从 0 到 (N-1) 进行逻辑编号，其中 N 是表空间中可用页的数目。

DMS 表空间中的页分组为扩展数据块（基于扩展数据块大小），并且从表空间管理的角度来看，所有对象分配都是以扩展数据块为基础完成的。即，表可能仅使用扩展数据块中的一半页，但是认为整个扩展数据块都在使用中，并且由该对象所有。缺省情况下，使用一个扩展数据块来存放容器标记，并且此扩展数据块中的页不能用来存放数据。但是，如果 DB2_USE_PAGE_CONTAINER_TAG 注册表变量已打开，则仅将一页用作容器标记。

下图显示了 DMS 表空间的逻辑地址映射。

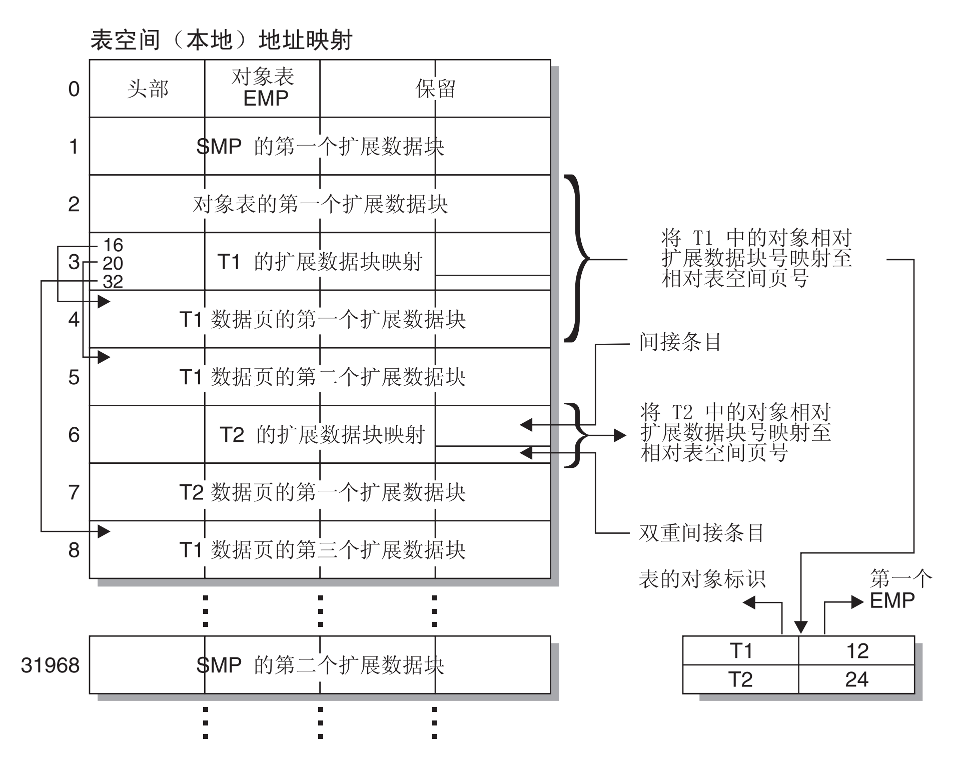


图 30. DMS 表空间

在表空间地址映射中，有两种类型的映射页：扩展数据块映像页（EMP）和空间映射页（SMP）。

对象表是一个内部关系表，它将对象标识映射至该表的第一个 EMP 扩展数据块的位置。这个 EMP 扩展数据块直接或间接地映射出该对象中的所有扩展数据块。每个 EMP 都包含一连串的条目。每一条目都将对象相对扩展数据块号映射至该对象扩展数据块所在的相对表空间页号。直接 EMP 条目直接将对象相对地址映射至相对表空间地址。第一个 EMP 扩展数据块中的最后一个 EMP 页包含间接条目。间接 EMP 条目映射至 EMP 页，EMP 页然后映射至对象页。第一个 EMP 扩展数据块中最后一个 EMP 页中的最后 16 条目包含双重间接条目。

逻辑表空间地址映射中的扩展数据块以循环顺序在与该表空间相关联的容器上进行条带分割。

因为容器中的空间是按扩展数据块来分配的，所以不会使用未组成完整扩展数据块的页。例如，如果您具有 205 页的容器，且扩展数据块大小为 10，则 1 个扩展数据块将用于存放标记，19 个扩展数据块将用于存放数据，并且剩余的 5 页将被浪费。

如果 DMS 表空间包含单个容器，则从逻辑页编号到磁盘上的物理位置的转换是直接进行的过程，其中 0、1 和 2 将以磁盘上的相同顺序定位。

当有多个容器并且每个容器大小相同时，它也是相当直接的过程。表空间（包含页 0 到（扩展数据块大小 - 1））中第一个扩展数据块将位于第一个容器中，第二个扩展数据块将位于第二个容器中，依此类推。在最后一个容器之后，过程将从第一个容器开始重复。这种循环过程可以保持数据平衡。

对于包含不同大小容器的表空间，不能使用每个容器轮流进行的简单方法，因为它不会利用大型容器中的额外空间。这就是引入表空间映射的位置 – 它规定如何在表空间内定位扩展数据块，确保物理容器中的所有扩展数据块都可用。

注: 在下列示例中，容器大小未将容器标记计算在内。容器大小很小，仅用于说明目的，它们不是建议的容器大小。这些示例显示表空间中不同大小的容器，但建议使用相同大小的容器。

示例 1:

表空间中有 3 个容器，每个容器包含 80 个可用的页，并且表空间的扩展数据块大小是 20。因此，每个容器有 4 个扩展数据块（80 / 20），总数为 12 个扩展数据块。这些扩展数据块位于磁盘上，如图 31 中所示。

表空间

容器 0	容器 1	容器 2
扩展数据块 0	扩展数据块 1	扩展数据块 2
扩展数据块 3	扩展数据块 4	扩展数据块 5
扩展数据块 6	扩展数据块 7	扩展数据块 8
扩展数据块 9	扩展数据块 10	扩展数据块 11

图 31. 带有三个容器和 12 个扩展数据块的表空间

要查看表空间映射，使用快照监视器获取表空间快照。在示例 1 中，三个容器大小相等，表空间映射为如下所示:

```
    范围     分割集          分割区      最大        最大       起始            结束        调节     容器
编号                        偏移     扩展数据块   页         分割区          分割区
[0]      [0]                0        11          239        0               3           0      3（0、1 和 2）
```

*范围*是其中连续范围的分割区包含同一组容器的一段映射。在示例 1 中，所有的分割区（0 到 3）都包含同一组 3 个容器（0、1 和 2），因此认为它是单个范围。

表空间映射中的标题是“范围编号”、“分割集”、“分割区偏移”、“根据范围寻址的最大扩展数据块编号”、“根据范围寻址的最大页编号”、“起始分割区”、“结束分割区”、“范围调节”和“容器列表”。这些将会在示例 2 中更详细地加以说明。

还可以对此表空间进行图解（如图 32 中所示），其中每条竖线对应一个容器，每条横线都称为分割区，并且每个单元编号对应一个扩展数据块。

容器

分割区	0	1	2
0	扩展数据块 0	扩展数据块 1	扩展数据块 2
1	扩展数据块 3	扩展数据块 4	扩展数据块 5
2	扩展数据块 6	扩展数据块 7	扩展数据块 8
3	扩展数据块 9	扩展数据块 10	扩展数据块 11

图 32. 带有三个容器和 12 个扩展数据块，突出显示分割区的表空间

示例 2:

表空间中有两个容器：第一个大小是 100 页，第二个大小是 50 页并且扩展数据块大小是 25。这意味着第一个容器具有四个扩展数据块并且第二个容器具有两个扩展数据块。可以对表空间进行图解，如图 33 中所示。

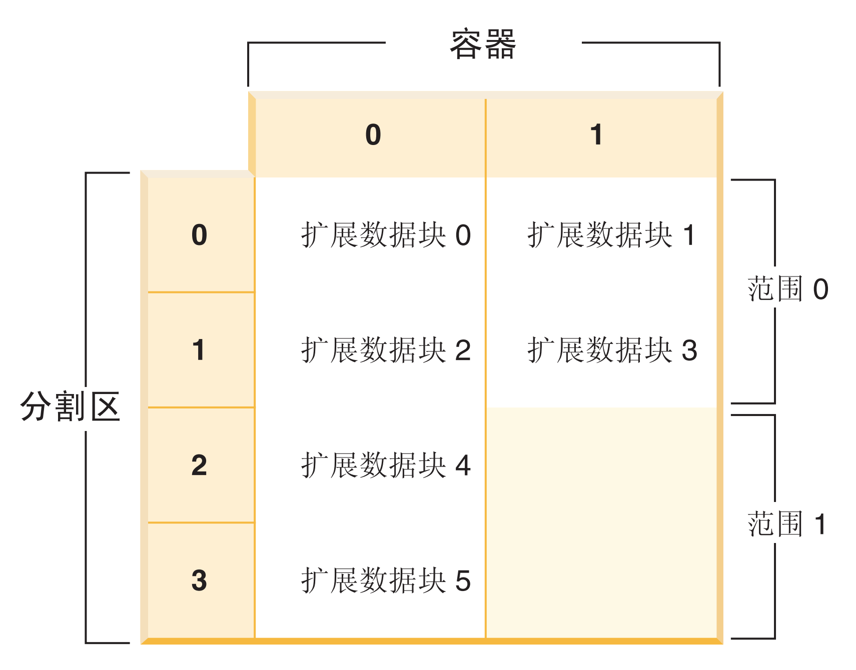


图 33. 带有两个容器，突出显示范围的表空间

分割区 0 和 1 包含两个容器（0 和 1），但是分割区 2 和 3 只包含第一个容器（0）。这些分割集的每一个分割集都是一个范围。表空间映射，如表空间快照中所示，类似如下:

```
    范围    分割集          分割区     最大        最大       起始            结束        调节       容器
 编号                      偏移    扩展数据块  页         分割区          分割区
  [0]     [0]               0          3        99         0               1          0        2 (0 和 1)
  [1]     [0]               0          5        149        2               3          0        1 (0)
```

在第一个范围中有四个扩展数据块，因此在此范围中寻址的最大扩展数据块编号（最大扩展数据块）为 3。每个扩展数据块有 25 页，因此在第一个范围中有 100 页。因为页的最大编号也是从 0 开始，所以此范围中寻址的最大页编号（最大页）是 99。此范围中的第一个分割区（起始分割区）是 0 并且最后一个分割区（结束分割区）是 1。此范围中有两个容器并且它们是 0 和 1。分割区偏移是分割集中的第一个分割，因为只有一个分割集，所以在此情况下它是 0。范围调节（Adj.）是在表空间中重新平衡数据时使用的偏移量。（当在表空间中添加或删除空间时会发生重新平衡。）当没有重新平衡时，它将始终为 0。

在第二个范围中有两个扩展数据块并且因为先前范围中寻址的最大扩展数据块编号是 3，所以此页中寻址的最大扩展数据块编号是 5。在第二个范围中有 50 页（2 个扩展数据块 * 25 页）并且由于在先前范围中寻址的最大页编号是 99，所以在此范围中寻址的最大页编号是 149。此范围从分割区 2 开始并且在分割区 3 结束。

相关概念:

- 『快照监视器』（*《系统监视器指南和参考》*）
- 第 110 页的『数据库管理的空间』
- 第 119 页的『在 DMS 表空间中如何添加和扩展容器』
- 第 128 页的『如何在 DMS 表空间中删除和缩小容器』

相关参考:

- 『GET SNAPSHOT command』（*Command Reference*）

在 DMS 表空间中如何添加和扩展容器

创建表空间时，会创建其表空间映射并对齐所有初始容器，以使它们都从分割区 0 开始。这意味着数据将均匀分布在所有表空间容器上，直到个别容器已满。（请参阅第 120 页的示例 1。）

ALTER TABLESPACE 语句允许向现有表空间添加容器或扩展容器，以增加其存储容量。

添加比现有容器小的容器会导致数据分布不均匀。这可能导致并行 I/O 操作（如预取数据）的执行效率比对大小相同的容器执行的效率要低。

向表空间添加新容器或扩展现有容器时，*可能*发生表空间数据的重新平衡。

重新平衡

添加或扩展容器时的在平衡过程涉及将表空间扩展数据块从一个位置移动到另一位置，这是通过试图保持数据在表空间内成为分割区来完成的。

在重新平衡期间不会限制对表空间的访问；可以像平常一样删除、创建、填充和查询对象。但是，重新平衡操作可能对性能有很大的影响。如果需要添加多个容器，并且计划重新平衡容器，则应在单个 ALTER TABLESPACE 语句中同时添加它们，以免数据库管理器不得不多次重新平衡数据。

表空间*高水位标记*在重新平衡过程中起着关键作用。高水位标记是表空间中分配的最高页的页数。例如，表空间有 1000 页，扩展数据块大小为 10，则结果为 100 个扩展数据块。如果第 42 个扩展数据块是表空间中最高分配的扩展数据块，则高水位标记是 42 * 10 = 420 页。这与已使用的页不同，因为可能已经释放了高水位标记下的一些扩展数据块，所以它们可供复用。

在重新平衡启动之前，会根据所作的容器更改构建新的表空间映射。重新平衡程序将扩展数据块从由当前映射确定的位置移至由新映射确定的位置。重新平衡程序从扩展数据块 0 开始，一次移动一个扩展数据块，直到移动了持有高水位标记的扩展数据块为止。移动每个扩展数据块时，当前映射每次改变一块以使其看起来与新映射相似。当完成重新平衡时，对于当前映射和新映射，一直到持有高水位标记的分割区，看起来应完全相同。于是使当前映射与新映射看起来完全相同，重新平衡过程就完成了。如果扩展数据块在当前映射中的位置与它在新映射中的位置相同，则不移动该扩展数据块，并且不发生 I/O。

当添加新容器时，该容器在新映射内的位置取决于其大小及其分割集中其他容器的大小。若容器足够大，以至于它可以从分割集中的第一个分割区开始，并在分割集中的最后一个分割区处（或以外）结束，则将它使用该方法放置（请参阅第 121 页的示例 2）。若容器不够大，无法做到这一点，则它将在映射中定位为在分割集的最后一个分割区结束（请参阅第 124 页的示例 4。）这样做是为了最小化需要重新平衡的数据量。

注: 在下列示例中，容器大小未将容器标记计算在内。容器大小很小，仅用于说明目的，它们不是建议的容器大小。这些示例显示表空间中不同大小的容器，但建议使用相同大小的容器。

示例 1:

如果您创建的表空间有三个容器，扩展数据块大小为 10，并且容器分别为 60、40 和 80 页（6、4 和 8 个扩展数据块），则将创建带有映射的表空间，可进行如第 121 页的图 34 中所示的图解。

分割区 \ 容器	0	1	2
0	扩展数据块 0	扩展数据块 1	扩展数据块 2
1	扩展数据块 3	扩展数据块 4	扩展数据块 5
2	扩展数据块 6	扩展数据块 7	扩展数据块 8
3	扩展数据块 9	扩展数据块 10	扩展数据块 11
4	扩展数据块 12		扩展数据块 13
5	扩展数据块 14		扩展数据块 15
6			扩展数据块 16
7			扩展数据块 17

图 34. 带有三个容器和 18 个扩展数据块的表空间

相应的表空间映射，如表空间快照中所示，类似如下:

范围编号	分割集	分割区偏移	最大扩展数据块	最大页	起始分割区	结束分割区	调节	容器
[0]	[0]	0	11	119	0	3	0	3（1 和 2）
[1]	[0]	0	15	159	4	5	0	2（0 和 2）
[2]	[0]	0	17	179	6	7	0	1（2）

表空间映射中的标题是“范围编号”、“分割集”、“分割区偏移”、“根据范围寻址的最大扩展数据块编号”、“根据范围寻址的最大页编号”、“起始分割区”、“结束分割区”、“范围调节”和“容器列表”。

示例 2:

如果在示例 1 中向表空间添加了一个 80 页的容器，容器就大到足以从第一个分割区（分割区 0）中开始，并在最后一个分割区（分割区 7）中结束。它被定位为从第一个分割区中开始。结果表空间可进行如图 35 中所示的图解。

分割区 \ 容器	0	1	2	3
0	扩展数据块 0	扩展数据块 1	扩展数据块 2	扩展数据块 3
1	扩展数据块 4	扩展数据块 5	扩展数据块 6	扩展数据块 7
2	扩展数据块 8	扩展数据块 9	扩展数据块 10	扩展数据块 11
3	扩展数据块 12	扩展数据块 13	扩展数据块 14	扩展数据块 15
4	扩展数据块 16		扩展数据块 17	扩展数据块 18
5	扩展数据块 19		扩展数据块 20	扩展数据块 21
6			扩展数据块 22	扩展数据块 23
7			扩展数据块 24	扩展数据块 25

图 35. 带有四个容器和 26 个扩展数据块的表空间

相应的表空间映射，如表空间快照中所示，类似如下：

```
    范围      分割集       分割区   最大       最大         起始         结束     调节  容器
编号                     偏移   扩展数据块   页          分割区       分割区
 [0]     [0]            0       15       159         0          3     0     4（0、1、2 和 3）
 [1]     [0]            0       21       219         4          5     0     3（0、2 和 3）
 [2]     [0]            0       25       259         6          7     0     2（2 和 3）
```

如果高水位标记在扩展数据块 14 以内，则重新平衡程序将从扩展数据块 0 开始，并且将把所有扩展数据块上移至 14（包括 14）。两个映射内的扩展数据块 0 的位置相同，所以不必移动此扩展数据块。扩展数据块 1 和 2 的情况相同。需要移动扩展数据块 3，所以从旧位置（容器 0 内的第二个扩展数据块）读取该扩展数据块并写至新位置（容器 3 内的第一个扩展数据块）。将移动此扩展数据块之后直到扩展数据块 14（包括扩展数据块 14）的每个扩展数据块。一旦移动了扩展数据块 14，当前映射看起来会像新映射，并且重新平衡程序将终止。

如果改变映射以使所有新添加的空间都在上限之后，则不需要重新平衡并且所有的空间都立即可用。如果改变映射以使部分空间在高水位标记之后，则分割区中在高水位标记之上的空间将是可用的。余下部分直到重新平衡完成才可用。

如果决定扩展容器，则重新平衡程序的功能相似。如果扩展容器以使它超出了分割集中的最后一个分割，则将扩展该分割集以适应这种情况并将相应地移出其后的分割集。结果是容器不会扩展到其后的任何分割集中。

示例 3:

以“示例 1”中的表空间为例。如果将容器 1 从 40 页扩展到 80 页，则新表空间将类似图 36。

分割区 \ 容器	0	1	2
0	扩展数据块 0	扩展数据块 1	扩展数据块 2
1	扩展数据块 3	扩展数据块 4	扩展数据块 5
2	扩展数据块 6	扩展数据块 7	扩展数据块 8
3	扩展数据块 9	扩展数据块 10	扩展数据块 11
4	扩展数据块 12	扩展数据块 13	扩展数据块 14
5	扩展数据块 15	扩展数据块 16	扩展数据块 17
6		扩展数据块 18	扩展数据块 19
7		扩展数据块 20	扩展数据块 21

图 36. 带有三个容器和 22 个扩展数据块的表空间

相应的表空间映射，如表空间快照中所示，类似如下:

```
    范围     分割集      分割区  最大     最大       起始       结束   调节 容器
编号                   偏移  扩展数据块  页        分割区      分割区
 [0]    [0]           0      17     179        0          5     0   3（0、1 和 2）
 [1]    [0]           0      21     219        6          7     0   2（1 和 2）
```

示例 4:

考虑第 120 页的示例 1 中的表空间。若向它添加一个 50 页（5 个扩展数据块）的容器，则将以如下方式将该容器添加至新映射。容器大小不足以从第一个分割区（分割区 0）中开始，并在最后一个分割区（分割区 7）处或以外结束，因此将它定位为在最后一个分割区中结束。（请参阅图 37。）

分割区 \ 容器	0	1	2	3
0	扩展数据块 0	扩展数据块 1	扩展数据块 2	
1	扩展数据块 3	扩展数据块 4	扩展数据块 5	
2	扩展数据块 6	扩展数据块 7	扩展数据块 8	
3	扩展数据块 9	扩展数据块 10	扩展数据块 11	扩展数据块 12
4	扩展数据块 13		扩展数据块 14	扩展数据块 15
5	扩展数据块 16		扩展数据块 17	扩展数据块 18
6			扩展数据块 19	扩展数据块 20
7			扩展数据块 21	扩展数据块 22

图 37. 带有四个容器和 23 个扩展数据块的表空间

相应的表空间映射，如表空间快照中所示，类似如下:

```
    范围     分割集      分割区  最大     最大       起始       结束   调节 容器
编号                   偏移  扩展数据块  页        分割区      分割区
[0]       [0]           0       8     89         0          2     0   3（0、1 和 2）
[1]       [0]           0      12    129         3          3     0   4（0、1、2 和 3）
[2]       [0]           0      18    189         4          5     0   3（0、2 和 3）
[3]       [0]           0      22    229         6          7     0   2（2 和 3）
```

要扩展容器，在 ALTER TABLESPACE 语句上使用 EXTEND 或 RESIZE 选项。要添加容器并重新平衡数据，在 ALTER TABLESPACE 语句上使用 ADD 选项。如果正在向已经有多个分割集的表空间添加容器，则可以指定想要向哪个分割集添加容器。为此，在 ALTER TABLESPACE 语句上使用 ADD TO STRIPE SET 选项。如果不指定分割集，则缺省行为将是向当前分割集添加容器。当前分割集是最新创建的分割集，而不是最后向其添加空间的分割集。

对分割集的任何更改可能导致对该分割集及其后的任何其他分割集的重新平衡。

可以通过使用表空间快照来监视重新平衡的进度。表空间快照可以提供关于重新平衡的信息，如重新平衡的开始时间、已经移动了多少个扩展数据块以及需要移动多少个扩展数据块。

没有重新平衡（使用分割集）

如果添加或扩展容器，并且添加的空间在表空间高水位标记之上，则不会发生重新平衡。

添加容器将总是在高水位标记下添加空间。换句话说，添加容器时，重新平衡通常是必要的。有一个选项可强制将新容器添加到高水位标记之上，它允许您选择不对表空间的内容重新平衡。此方法的一个优点是新容器可立即使用。不进行重新平衡这一选项仅在添加容器时才适用，在扩展现有容器时不适用。扩展容器时，仅当添加的空间在高水位标记之上时，才能避免重新平衡。例如，如果有许多大小相同的容器，并且按相同的量扩展它们，则扩展数据块的相对位置不会更改，并且不会发生重新平衡。

将容器添加到表空间中而不进行重新平衡可通过添加新的*分割集*来实现。分割集是表空间中的一组容器，数据在其上进行分割，且独立于属于该表空间的其他容器。现有分割集中的现有容器保持不变，而添加的容器成为新分割集的一部分。

要添加容器而不进行重新平衡，在 ALTER TABLESPACE 语句上使用 BEGIN NEW STRIPE SET 选项。

示例 5:

如果表空间有三个容器，扩展数据块大小为 10，并且容器分别为 30、40 和 40 页（分别为 3、4 和 4 个扩展数据块），则表空间可进行如第 126 页的图 38 中所示的图解。

容器

分割区

	0	1	2
0	扩展数据块 0	扩展数据块 1	扩展数据块 2
1	扩展数据块 3	扩展数据块 4	扩展数据块 5
2	扩展数据块 6	扩展数据块 7	扩展数据块 8
3		扩展数据块 9	扩展数据块 10

图 38. 带有三个容器和 11 个扩展数据块的表空间

相应的表空间映射，如表空间快照中所示，类似如下:

```
    范围      分割集        分割区   最大        最大          起始          结束     调节  容器
编号                     偏移   扩展数据块   页            分割区        分割区
 [0]     [0]              0         8      89             0             2      0    3（0、1 和 2）
 [1]     [0]              0        10     109             3             3      0    2（1 和 2）
```

示例 6:

在使用 BEGIN NEW STRIPE SET 选项添加 30 页和 40 页的两个新容器（分别为 3 和 4 个扩展数据块）时，不会影响现有范围；而是将创建一组新范围。这一组新范围是一个分割集，而最新创建的分割集称为当前分割集。添加了两个新的容器之后，表空间将类似第 127 页的图 39。



图 39. 带有两个分割集的表空间

相应的表空间映射，如表空间快照中所示，类似如下:

```
     范围      分割集       分割区   最大        最大          起始          结束    调节  容器
编号                     偏移   扩展数据块   页           分割区        分割区
 [0]     [0]              0        8      89            0             2      0    3（0、1 和 2）
 [1]     [0]              0       10     109            3             3      0    2（1 和 2）
 [2]     [1]              4       16     169            4             6      0    2（3 和 4）
 [3]     [1]              4       17     179            7             7      0    1（4）
```

如果向表空间添加新的容器，并且未将 TO STRIPE SET 选项与 ADD 子句配合使用，则将把容器添加至当前分割集（最高分割集）。可以使用 ADD TO STRIPE SET 子句来将容器添加到表空间中的任何分割集。必须指定有效的分割集。

DB2 数据库 Linux 版、UNIX 版和 Windows 版使用表空间映射来跟踪分割集，并且添加新容器而不进行重新平衡通常将导致映射比对容器进行重新平衡时增长得快。当表空间映射变得过大时，如果试图添加更多容器，将接收到错误 SQL0259N。

相关概念:

- 第 115 页的『表空间映射』

相关任务:

- 『将容器添加至 DMS 表空间』（*《管理指南: 实施》*）
- 『修改 DMS 表空间中的容器』（*《管理指南: 实施》*）

相关参考:

- 『表空间活动监控器元素』（*《系统监视器指南和参考》*）
- 『ALTER TABLESPACE statement』（*SQL Reference, Volume 2*）

- 『GET SNAPSHOT command』(*Command Reference*)

如何在 DMS 表空间中删除和缩小容器

对于 DMS 表空间，可以从表空间中删除容器或缩小容器的大小。使用 ALTER TABLESPACE 语句来完成此操作。

仅当该操作删除的扩展数据块的数目小于或等于表空间中的高水位标记之上的可用扩展数据块的数目时，才允许删除或缩小容器。这是必须的，因为该操作不能更改页号，从而直到高水位标记（包括高水位标记）的所有扩展数据块在表空间内必须处于相同的逻辑位置。因此，结果表空间必须具有足够的空间才能存放直到高水位标记（包括高水位标记）的所有数据。在没有足够的可用空间的情况下，执行语句时会立即接收到错误。

高水位标记是表空间中分配的最高页的页数。例如，表空间有 1000 页，扩展数据块大小为 10，则结果为 100 个扩展数据块。如果第 42 个扩展数据块是表空间中最高分配的扩展数据块，则意味着高水位标记是 42 * 10 = 420 页。这与已使用的页不同，因为可能已经释放了高水位标记下的一些扩展数据块，所以它们可供复用。

删除或缩小容器时，如果数据位于正从表空间中删除的空间中，则将进行重新平衡。在重新平衡启动之前，会根据所作的容器更改构建新的表空间映射。重新平衡程序将把扩展数据块从由当前映射确定的位置移至由新映射确定的位置。重新平衡程序从包含高水位标记的扩展数据块开始，一次移动一个扩展数据块，直到移动了扩展数据块 0 为止。移动每个扩展数据块时，当前映射每次改变一块以使其看起来与新映射相似。如果扩展数据块在当前映射中的位置与它在新映射中的位置相同，则不移动该扩展数据块，并且不发生 I/O。因为重新平衡从表空间中分配的最高扩展数据块开始移动，并以表空间中的第一个扩展数据块结束，所以又称为*逆向重新平衡*（与在添加或扩展容器之后向表空间添加空间时发生的*正向重新平衡*相反）。

删除容器时，余下的容器将重新编号，它们的容器标识从 0 开始每次加 1。如果删除了分割集中的所有容器，则将从映射除去该分割集，并且会下移映射中其后的所有分割集并对其重新编号，以便分割集号码中没有间隔。

注: 在下列示例中，容器大小未将容器标记计算在内。容器大小很小，仅用于说明目的，它们不是建议的容器大小。这些示例显示表空间中不同大小的容器，但是这只是用于说明目的；建议使用相同大小的容器。

例如，考虑这样一个表空间，它有三个容器，扩展数据块大小为 10。容器分别为 20、50 和 50 页（2、5 和 5 个扩展数据块）。表空间的图表显示在第 129 页的图 40 中。

容器

分割区	0	1	2
0	扩展数据块 0	扩展数据块 1	扩展数据块 2
1	扩展数据块 3	扩展数据块 4	扩展数据块 5
2		扩展数据块 6	扩展数据块 7
3		x	x
4		x	x

图 40. 带有 12 个扩展数据块（包括四个无数据的扩展数据块）的表空间

X 指示存在扩展数据块，但其中没有数据。

如果想要删除有两个扩展数据块的容器 0，则高水位标记之上必须至少具有两个可用扩展数据块。高水位标记在扩展数据块 7 中，有四个可用扩展数据块，因此可以删除容器 0。

相应的表空间映射，如表空间快照中所示，类似如下：

```
    范围     分割集         分割区     最大        最大         起始             结束   调节    容器
编号                    偏移    扩展数据块    页        分割区           分割区
[0]      [0]            0         5       59          0               1      0     3（0、1 和 2）
[1]      [0]            0        11      119          2               4      0     2（1 和 2）
```

删除之后，表空间将仅有容器 0 和容器 1。新的表空间图表显示在第 130 页的图 41 中。

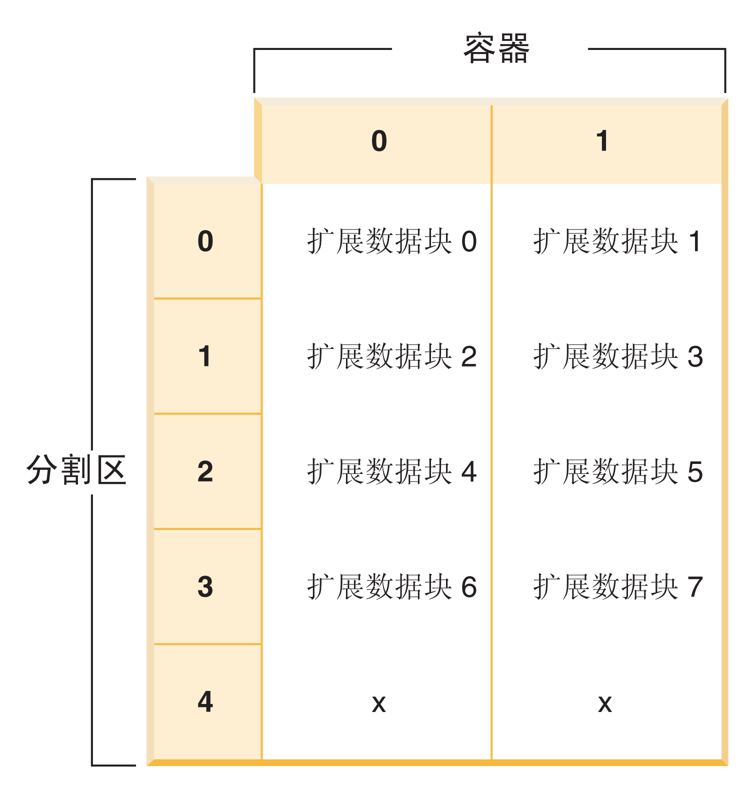


图 41. 删除容器之后的表空间

相应的表空间映射，如表空间快照中所示，类似如下：

```
      范围      分割集           分割区     最大         最大        起始          结束     调节  容器
编号                         偏移     扩展数据块      页         分割区       分割区
[0]       [0]                  0          9        99         0            4      0   2 (0 和 1)
```

如果想要缩小容器，重新平衡程序以相似的方式工作。

要缩小容器，在 ALTER TABLESPACE 语句上使用 REDUCE 或 RESIZE 选项。要删除容器，在 ALTER TABLESPACE 语句上使用 DROP 选项。

相关概念:

- 第 115 页的『表空间映射』

相关任务:

- 『修改 DMS 表空间中的容器』（*《管理指南：实施》*）

相关参考:

- 『ALTER TABLESPACE statement』（*SQL Reference, Volume 2*）
- 『GET SNAPSHOT command』（*Command Reference*）
- 『表空间活动监控器元素』（*《系统监视器指南和参考》*）

SMS 和 DMS 表空间的比较

在确定应使用哪种类型的表空间来存储数据时，需要考虑一些问题。

SMS 表空间的优点:

- 直到有需要时，系统才会分配空间。
- 由于不必预定义容器，所以创建表空间需要的初始工作较少。
- 对分布式数据创建的索引可以与表数据存储在不同的表空间中。

DMS 表空间的优点:

- 通过使用 ALTER TABLESPACE 语句，可添加或扩展容器来增加表空间的大小。现有数据可以自动在新的容器集合中重新平衡，以保持最优 I/O 效率。
- 根据存储的数据的类型，一个表可以分布在多个表空间中:
  - 长型字段和 LOB 数据
  - 索引
  - 规则表数据

  通过分隔表数据，可以提高性能或增加每个表存储的数据量。例如，一个表可以有 64GB 正规表数据、64GB 索引数据和 2TB 长型数据。若使用的是 8 KB 页，则表数据和索引数据可达 128 GB。若使用的是 16 KB 页，则表数据和索引数据可达 256 GB。若使用的是 32 KB 页，则表数据和索引数据可达 512 GB。
- 对分布式数据创建的索引可以与表数据存储在不同的表空间中。
- 可控制数据在磁盘上的位置（若操作系统允许的话）。
- 若所有的表数据都位于单个表空间中，可以使用比删除和重新定义一个表更少的开销来删除和重新定义一个表空间。
- 通常，精心调整的一组 DMS 表空间的性能将优于 SMS 表空间。

注:

1. 在 Solaris 操作系统上，对于性能要求高的工作负载，强烈建议使用具有原始设备的 DMS 表空间。
2. 对于性能敏感的应用程序，特别是涉及大量插入操作的应用程序，建议您使用 DMS 表空间。

并且，在这两种类型的表空间上，数据的放置也会有所不同。例如，考虑高效表扫描的这样的一个需要：重要的是扩展数据块中的页在物理上是连续的。对于 SMS，操作系统的文件系统决定了每个逻辑文件页的物理放置位置。根据文件系统上其他活动的级别以及用来确定放置位置的算法的不同，可能会连续分配这些页。但是，对于 DMS，因为数据库管理器直接与磁盘打交道，所以它可以确保这些页在物理上是连续的。

通常，小型个人数据库用 SMS 表空间管理最容易。另一方面，对于大的、增长中的数据库，您可能只希望使用 SMS 表空间用作临时表空间和目录表空间，而使用具有多个容器的单独的 DMS 表空间用于每个表。另外，您可能想将长型字段数据和索引存储在它们自己的表空间中。

如果选择使用带设备容器的 DMS 表空间，则需要调整和管理您的环境。

相关概念:

• 第 110 页的『数据库管理的空间』
• 第 107 页的『系统管理的空间』
• 第 102 页的『表空间设计』

表空间磁盘 I/O

表空间的类型和设计决定了对该表空间执行的 I/O 的效率。在进一步考虑关于表空间的设计和使用的问题之前，您应了解下列概念。

大块读取
在单个请求中检索多页（通常为一个扩展数据块）的一种读取。一次读取几页比分别读取每页更有效。

预取
在一个查询引用页之前对那些页的读取。总的目的是为缩短响应时间。若页的预取可以与查询的执行异步发生，就能够达到此目的。当 CPU 或 I/O 子系统以最大能力运行时达到最佳响应时间。

页清除
当读取和修改页时，它们会累积在数据库缓冲池中。当读入一页时，便将其读入到缓冲池页中。若该缓冲池已充满修改的页，则必须将修改的这些页的其中一页写出至磁盘，然后才能再读入新的页。为避免缓冲池变满，页清除程序代理程序的任务就是写出修改的页，以保证缓冲池页可用于将来的读取请求。

无论何时，只要是有利，DB2 数据库 Linux 版、UNIX 版和 Windows 版就会执行大块读取。当检索顺序排列或本质上是部分顺序排列的数据时，通常会发生这种情况。在一个读取操作中读取的数据量取决于扩展数据块大小 － 扩展数据块大小越大，一次可以读取的页就越多。

如果可以将页从磁盘读入缓冲池内的连续页中，则可以进一步提高顺序预取的性能。因为缺省情况下缓冲池是基于页的，所以从磁盘的连续页中读取时，不能保证会找到一组连续页。基于块的缓冲池可以用于此目的，因为它们不仅包含页区域，还包含可用于几组连续页的块区域。每一组连续页都命名为一个块，并且每个块都包含称为块大小的若干页。页和块区域的大小以及每个块中的页的数目都是可配置的。

扩展数据块存储在磁盘上的方式影响 I/O 效率。在使用设备容器的 DMS 表空间中，数据往往在磁盘上是连续的，且可以在最短的搜索时间和磁盘等待时间内进行读取。若使用的是文件，则为了供 DMS 表空间使用而预分配的大文件在磁盘上也往往是连续的，尤其当该文件分配在一个干净的文件空间中时更是这样。但是，数据可能被系统文件分散，并存储在磁盘上的多个位置中。当使用一次将文件扩展一页的 SMS 表空间时（这使产生碎片的概率更高），最可能发生此情况。

可以通过更改 CREATE TABLESPACE 或 ALTER TABLESPACE 语句上的 PREFETCHSIZE 选项来控制预取的程度。（该数据库中所有表空间的缺省值由 *dft_prefetch_sz* 数据库配置参数设置。）PREFETCHSIZE 参数告诉 DB2 在触发预取时要读取的页数。通过在 CREATE TABLESPACE 语句上将 PREFETCHSIZE 设置为 EXTENTSIZE 参数的倍数，可以并行读取多个扩展数据块。（该数据库中所有表空间的缺省值由 *dft_extent_sz* 数据库配置参数设置。）EXTENTSIZE 参数指定在跳至下一个容器之前将写入一个容器的 4 KB 大小的页数。

例如，假定有一个表空间使用三个设备。若将 PREFETCHSIZE 设置为 EXTENTSIZE 的三倍，则 DB2 可以用并行方式从每个设备中执行大块读取，从而显著增加 I/O 吞吐

量。此情况假定每个设备是一个单独的物理设备，且控制器具有足够的带宽来处理来自每个设备的数据流。注意，DB2 可能需要根据查询速度、缓冲池的利用情况和其他因素在运行时动态调整预取参数。

某些文件系统使用它们自己的预取方法（例如，在 AIX 上的“日志文件系统”）。在某些情况中，文件系统的预取设置得比 DB2 的预取更主动。这可能导致使用文件容器的 SMS 和 DMS 表空间的预取似乎比使用设备的 DMS 表空间的预取执行效率更高。但这是误导，因为它可能是在文件系统中发生的其他级别的预取的结果。DMS 表空间应该能够比任何等效配置的执行效率高。

为提高预取效率（甚至读取效率），必须存在足够数量的干净缓冲池页。例如，可能有一个并行预取请求要从一个表空间读取三个扩展数据块，对于正在读取的每一页，从缓冲池写出经过修改的一页。该预取请求可能被拖慢，导致它跟不上查询的进展。应配置足够数量的页清除程序，以满足预取请求。

相关概念:

- 『将数据预取至缓冲池』（*《性能指南》*）
- 第 102 页的『表空间设计』

相关参考:

- 『ALTER TABLESPACE statement』（*SQL Reference, Volume 2*）
- 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）

表空间设计中的工作负载注意事项

在您的环境中由 DB2 数据库 Linux 版、UNIX 版和 Windows 版管理的主要工作负载类型会影响您选择要使用的表空间类型以及要指定的页大小。联机事务处理（OLTP）工作负载的特征是：事务需要对数据进行随机访问，通常涉及频繁插入或更新活动和通常返回一小组数据的查询。假定访问是随机的，并且是访问一页或几页，则不太可能发生预取。

使用设备容器的 DMS 表空间在这种情况下表现得最好。若不需要最大性能，使用文件容器的 DMS 表空间或 SMS 表空间也适合用于 OLTP 工作负载。若期望很少的顺序 I/O 或不期望它，则 CREATE TABLESPACE 语句上的 EXTENTSIZE 和 PREFETCHSIZE 参数的设置对于 I/O 的效率就显得不重要。但是，使用 *chngpgs_thresh* 配置参数设置足够数目的页清除程序是很重要的。

查询工作负载的特征是，事务需要对数据进行顺序访问或部分顺序访问，并常常返回大的数据集。使用多个设备容器（每个容器都在单独的磁盘上）的 DMS 表空间，最有可能提供有效的并行预取。应该将 CREATE TABLESPACE 语句上的 PREFETCHSIZE 参数的值设置为 EXTENTSIZE 参数的值乘以设备容器数之积。这允许 DB2 以并行方式从所有容器中预取。如果容器的数目更改，或需要使预取更多或更少，则可以使用 ALTER TABLESPACE 语句相应地更改 PREFETCHSIZE 值。

查询工作负载的一个合理的替代方法是使用文件（若文件系统有自己的预取的话）。这些文件可以是使用文件容器的 DMS 类型或 SMS 类型。注意，若使用 SMS，则需要将目录容器映射至单独的物理磁盘，以实现 I/O 并行性。

混合工作负载的目标是：对于 OLTP 工作负载，使单个 I/O 请求尽可能有效率；而对于查询工作负载，最大程度地提高并行 I/O 的效率。

确定表空间页大小的注意事项如下所示：

- 对于执行随机行读写操作的 OLTP 应用程序，通常最好使用较小的页大小，这样不需要的行就不会浪费缓冲池空间。
- 对于一次访问大量连续行的决策支持系统（DSS）应用程序，页大小大一些会比较好，这样就能减少读取特定数目的行所需的 I/O 请求数。但是，也有例外。如果行大小小于：

  ```
  页大小 / 255
  ```

  则每页上都会有浪费的空间（每页最多有 255 行）。在这种情况下，更小一点的页大小可能更合适。
- 更大的页大小可允许您减少索引中的级别数。
- 越大的页，支持的行越长。
- 在缺省的 4 KB 页上，一个表只能有 500 列，而更大的页大小（8 KB、16 KB 和 32 KB）支持 1012 列。
- 表空间的最大大小与表空间的页大小成正比。

相关概念：

- 第 110 页的『数据库管理的空间』
- 第 107 页的『系统管理的空间』

相关参考：

- 『ALTER TABLESPACE statement』（*SQL Reference, Volume 2*）
- 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）
- 『SQL and XQuery limits』（*SQL Reference, Volume 1*）
- 『chngpgs_thresh － 已更改页数的阈值配置参数』（*《性能指南》*）

扩展数据块大小

表空间的扩展数据块大小表示在将数据写入下一个容器之前，将写入当前容器的表数据的页数。当选择扩展数据块大小时，应考虑：

- 表空间中表的大小和类型。

  将 DMS 表空间中的空间一次分配给表一个扩展数据块。当填充该表而一个扩展数据块变满时，会分配新的扩展数据块。保留了 DMS 表空间容器存储器，这意味着将分配新的扩展数据块，直到彻底用完容器为止。

  将 SMS 表空间中的空间一次分配给表一个扩展数据块或者一次分配给表一页。当填充该表而一个扩展数据块或页变满时，会分配新的扩展数据块或页，直到使用了文件系统中的所有扩展数据块或页为止。当使用 SMS 表空间时，允许进行多页文件分配。多页文件分配允许分配扩展数据块而不是一次分配一页。

  缺省情况下，启用了多页文件分配功能。multipage_alloc 数据库配置参数值将指示是否已启用多页文件分配功能。

  注： 多页文件分配功能不适用于临时表空间。

一个表由下列单独的表对象组成:

– 数据对象。它是存储规则列数据的地方。
– 索引对象。在表上定义的所有索引都存储在这里。
– 长型字段对象。若表有一个或多个 LONG 列，则长型字段数据存储在此处。
– 两个 LOB 对象。若表有一个或多个 LOB 列，则它们都存储在这两个表对象中:
  - 一个表对象用于存储 LOB 数据
  - 第二个表对象用于存储描述 LOB 数据的元数据
– 多维表的块映射对象。

每个表对象都是单独存储的，每个对象按需要分配新的扩展数据块。每个 DMS 表对象还与称为*扩展数据块映像*的元数据对象配成一对，该元数据对象描述该表空间中属于该表对象的所有扩展数据块。用于扩展数据块映像的空间也是以一次一个扩展数据块的方式分配。

因此，DMS 表空间中对象的初始空间分配是两个扩展数据块。（SMS 表空间中对象的初始空间分配是一页。）因此，若您在一个 DMS 表空间中有多个较小的表，您可能要分配相对大的空间来存储相对少量的数据。在这种情况下，应该指定小的扩展数据块大小。

否则，若您有一个增长速率高的非常大的表，且您使用具有较小扩展数据块大小的 DMS 表空间，您可能会有与其他扩展数据块的频繁分配相关的不需要的开销。

• 对这些表访问的类型。

  若对表的访问包括许多查询或处理大量数据的事务，则从表中预取数据可以显著改善性能。

• 必需的扩展数据块的最小数目。

  若容器中没有足够的空间以供表空间的五个扩展数据块使用，则将无法创建表空间。

相关概念:

• 第 102 页的『表空间设计』

相关参考:

• 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）
• 『db2empfa - Enable multipage file allocation command』（*Command Reference*）
• 『multipage_alloc － 启用的多页文件分配配置参数』（*《性能指南》*）

表空间和缓冲池之间的关系

每个表空间都与一个特定的缓冲池相关。缺省缓冲池是 IBMDEFAULTBP。若另一个缓冲池要与表空间相关，则该缓冲池必须存在（用 CREATE BUFFERPOOL 语句定义它），它必须具有相同的页大小，且在创建该表空间（使用 CREATE TABLESPACE 语句）时定义关联。表空间与缓冲池之间的关联可以使用 ALTER TABLESPACE 语句来更改。

若拥有多个缓冲池，则可以配置数据库使用的内存，以改善整体性能。例如，如果具有带有一个或多个用户可随机访问的大（大于可用内存）表的表空间，缓冲池的大小

可能受到限制，因为高速缓存该数据页可能没有好处。用于联机事务应用程序的表空间可以与一个较大的缓冲池相关联，以便可以更长时间地高速缓存应用程序所使用的数据页，导致响应时间更快。配置新缓冲池时必须小心。

注: 若确定数据库需要 8 KB、16 KB 或 32 KB 的页大小，必须将使用其中一种页大小的每个表空间映射至具有相同页大小的缓冲池。

当启动数据库时，所有缓冲池必需的存储器对于数据库管理器必须是可用的。如果 DB2 数据库 Linux 版、UNIX 版和 Windows 版无法获取必需的存储器，则数据库管理器将使用缺省缓冲池（具有 4 KB、8 KB、16 KB 和 32 KB 页大小的缓冲池各一个）启动，并发出警告。

在分区数据库环境中，可以为该数据库中的所有数据库分区创建一个大小相同的缓冲池。也可以在不同数据库分区上创建不同大小的缓冲池。

相关概念:

- 『Table spaces and other storage structures』（*SQL Reference, Volume 1*）

相关参考:

- 『ALTER BUFFERPOOL statement』（*SQL Reference, Volume 2*）
- 『ALTER TABLESPACE statement』（*SQL Reference, Volume 2*）
- 『CREATE BUFFERPOOL statement』（*SQL Reference, Volume 2*）
- 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）

表空间和数据库分区组之间的关系

在分区数据库环境中，每个表空间都与特定数据库分区组相关。这允许将表空间的特征应用于该数据库分区组中的每个数据库分区。该数据库分区组必须存在（用 CREATE DATABASE PARTITION GROUP 语句定义它），且当使用 CREATE TABLESPACE 语句创建表空间时定义表空间与该数据库分区组之间的关联。

不能使用 ALTER TABLESPACE 语句来更改表空间与数据库分区组之间的关联。只能为数据库分区组内的个别数据库分区更改表空间规范。在单分区环境中，每个表空间都与缺省数据库分区组相关。在定义表空间时，缺省数据库分区组为 IBMDEFAULTGROUP，除非在定义系统临时表空间，那时将使用 IBMTEMPGROUP。

相关概念:

- 『Table spaces and other storage structures』（*SQL Reference, Volume 1*）
- 第 78 页的『数据库分区组』
- 第 102 页的『表空间设计』

相关参考:

- 『CREATE DATABASE PARTITION GROUP statement』（*SQL Reference, Volume 2*）
- 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）

“存储管理”视图

使用“存储管理”视图来监视分区数据库的存储状态。“存储管理”视图是“存储管理”工具的图形界面。在“存储管理”视图中，可以获取数据库、数据库分区组或表空间的存储快照。当生成了表空间快照时，就从系统目录和在给定表空间作用域下定义的表、索引和容器的数据库监视器中收集统计信息。当生成了数据库或数据库分区组的快照时，就为在给定数据库或数据库分区组中定义的所有表空间收集统计信息。当生成了数据库快照时，就为数据库内的所有数据库分区组收集统计信息。可以使用不同类型的存储器快照来帮助您监视存储器的不同方面：

- 可以通过表空间的快照来监视空间使用率。
- 仅在分区数据库上：通过数据库分区组的快照，可以最好地监视数据偏差情况（数据库分布）。
- 可以通过数据库分区组快照和表空间快照来捕获索引的集群比率。索引的集群比率是通过索引文件夹的详细视图显示的。

“存储管理”视图还使您能够设置数据偏差阈值、空间使用率阈值和索引集群比率阈值。如果目标对象超过了指定的阈值，则在“存储管理”视图中，该对象及其父对象旁边的图标将用警告标志或警报标志来标记。

注： 只能为分区数据库设置数据偏差阈值。

使用“存储管理”启动板来指导您完成设置“存储管理”工具所需完成的任务。“存储管理”工具使您能够管理特定数据库或数据库分区的长期存储方式。它还允许您捕获数据分布快照以及查看存储历史记录。创建数据库时，将自动为存储管理工具创建三个存储过程函数：SYSPROC.CREATE_STORAGEMGMT_TABLES、SYSPROC.DROP_STORAGEMGMT_TABLES 和 SYSPROC.CAPTURE_STORAGEMGMT_INFO。将根据需要绑定它们相应的程序包。

注： 在控制中心中，可以从数据库、数据库分区组或表空间对象打开“存储管理设置”启动板。该启动板将引导您完成只需执行一次的设置过程，以便使用“存储管理”工具。使用“存储管理设置”启动板捕获所选对象或其父对象的快照后，就可以打开“存储管理”视图了。

相关参考：

- 『CAPTURE_STORAGEMGMT_INFO procedure – Retrieve storage-related information for a given root object』（*Administrative SQL Routines and Views*）
- 『CREATE_STORAGEMGMT_TABLES procedure – Create storage management tables』（*Administrative SQL Routines and Views*）
- 『DROP_STORAGEMGMT_TABLES procedure – Drop all storage management tables』（*Administrative SQL Routines and Views*）
- 第 138 页的『“存储管理”视图表』

存储管理工具的存储过程

下表显示为存储管理工具创建的存储过程功能。创建数据库时，就自动创建了存储过程。另外，将根据需要绑定它们相应的程序包。

表 24. 存储管理工具的存储过程

标准名称	参数	功能
SYSPROC.CREATE_STORAGEMGMT _TABLES	in_tbspace VARCHAR(128) input – 表空间名	在由 input 指定的表空间中以固定的“DB2TOOLS”模式创建所有的存储管理表。
SYSPROC.DROP_STORAGEMGMT _TABLES	dropSpec SMALLINT input - 0 / 1	试图删除所有存储管理表。当 dropSpec=0 时，当遇到任何错误时，进程将停止；当 dropSpec=1 时，进程将继续并忽略它遇到的任何错误。
SYSPROC.CAPTURE_STORAGEMGMT _INFO	in_rootType SMALLINT input – 所有有效值都是在 STMG_OBJECT_TYPE 表中给定的 in_rootSchema VARCHAR(128) input – 存储器快照根对象的模式名 in_rootName VARCHAR(128) input – 根对象的名称	试图对给定的根对象以及在它的作用域内定义的存储对象收集系统目录和快照的与存储器相关的信息。所有存储对象都是在 STMG_OBJECT_TYPE 表中指定的。

相关参考:

- 第 137 页的『“存储管理”视图』

“存储管理”视图表

STMG_OBJECT_TYPE 表:

对于每种受支持的可以监视的存储类型，STMG_OBJECT_TYPE 表中都包含一行。

必须将 STMG_OBJECT_TYPE 指定为 capture_storagemgmt_info() 存储过程的第一个参数。例如:

```
sysproc.capture_storagemgmt_info(<stmg_object_type>, <object_schema>, <object_name>)
```

第一个参数 stmg_object_type 是由下表中的条目定义的。

表 25. STMG_OBJECT_TYPE 表

列名	数据类型	可空	描述
OBJ_TYPE	INTEGER	N	与一种类型的存储对象相对应的整数值 0 – 数据库 1 – 数据库分区组 2 – 表空间 3 – 表空间容器 4 – 表 5 – 索引

表 25. STMG_OBJECT_TYPE 表 (续)

列名	数据类型	可空	描述
TYPE_NAME	VARCHAR	N	存储对象类型的描述名 STMG_DATABASE STMG_DBPGROUP STMG_TABLESPACE STMG_CONTAINER STMG_TABLE STMG_INDEX

STMG_THRESHOLD_REGISTRY 表:

对于每种存储阈值类型，STMG_THRESHOLD_REGISTRY 表中都包含一行。当生成了存储快照时，分析过程就将使用已启用的阈值。如果启用了阈值类型，则将对正在监视的数据执行阈值分析，超过阈值的列将用指定的阈值类型的适当值更新。

例如: :

要对表空间的空间用途禁用阈值分析:

```
db2 UPDATE SYSTOOLS.STMG_THRESHOLD_REGISTRY SET ENABLED = 'N'
   WHERE STMG_TH_TYPE = 1
```

表 26. STMG_THRESHOLD_REGISTRY 表

列名	数据类型	可空	描述
STMG_TH_TYPE	INTEGER	N	与存储阈值类型相对应的整数值 1 = STMG SPACE USAGE THRESHOLD 2 = STMG DATA SKEW THRESHOLD 3 = STMG CLUSTER RATIO THRESHOLD
ENABLED	CHARACTER	N	Y = 启用了阈值 N = 未启用阈值，因此在存储分析期间不会与它进行比较
STMG_TH_NAME	VARCHAR	Y	存储阈值的描述性名称 STMG CLUSTER RATIO THRESHOLD STMG SPACE USAGE THRESHOLD STMG DATA SKEW THRESHOLD

STMG_CURR_THRESHOLD 表:

对于为存储对象显式设置的每种阈值类型，STMG_CURR_THRESHOLD 表中都包含一行。执行新的存储快照时，如果对正在捕获的对象启用了阈值分析（请参见表 26），则此表中的值将用来确定对正在监视的每类阈值设置的警告和警报阈值。如果正在分析的对象没有在此表中明确设置阈值，则使用该对象类型的父代对象的阈值。缺省情

况下，此表包含三行，每行对应一个阈值类型。这三行中的阈值是对数据库对象以及数据库所有其他对象的父代设置的。包括在存储快照分析中的所有对象都将自动从数据库对象继承这些阈值，除非对子代对象明确设置了阈值，例如，表空间或表。

例如: :

要将数据库中所有对象的空间用途警告和警报阈值设置为 90 和 95:

```
db2 UPDATE SYSTOOLS.STMG_CURR_THRESHOLD SET WARNING_THRESHOLD = 90,
    ALARM_THRESHOLD = 95
    WHERE STMG_TH_TYPE = 1 AND OBJ_TYPE = 0
```

表 27. STMG_CURR_THRESHOLD 表

列名	数据类型	可空	描述
STMG_TH_TYPE	INTEGER	N	与存储阈值类型相对应的整数值。请参阅第 139 页的表 26以了解阈值类型的定义。
OBJ_TYPE	INTEGER	N	与存储对象的类型相对应的整数值。请参阅第 138 页的表 25以了解阈值类型的定义。
OBJ_NAME	VARCHAR	N	存储对象的名称。
OBJ_SCHEMA	VARCHAR	N	存储对象的模式。 当模式不适合于对象时，将使用 “-”
WARNING_THRESHOLD	SMALLINT	Y	为存储对象设置的警告阈值的值。
ALARM_THRESHOLD	SMALLINT	Y	为存储对象设置的警报阈值的值。

STMG_ROOT_OBJECT 表:

对于每个存储快照的根对象，STMG_ROOT_OBJECT 表中都包含一行。可以通过删除此表中的条目来删除完整的存储快照。

例如: :

1. 删除所有存储管理快照:

   ```
   db2 DELETE FROM SYSTOOLS.STMG_ROOT_OBJECT
   ```

2. 删除所有表空间快照:

   ```
   db2 DELETE FROM SYSTOOLS.STMG_ROOT_OBJECT WHERE OBJ_TYPE = 2
   ```

表 28. STMG_ROOT_OBJECT 表

列名	数据类型	可空	描述
STMG_TIMESTAMP	TIMESTAMP	N	存储快照的时间戳记。它指示数据捕获过程何时开始。
OBJ_TYPE	INTEGER	N	与存储对象的类型相对应的整数值。请参阅第 138 页的表 25以了解阈值类型的定义。
ROOT_ID	VARCHAR	N	根对象的标识。

STMG_OBJECT 表:

对于由迄今为止已生成的存储快照分析的每个存储对象，STMG_OBJECT 表中都包含一行。

注： 在列中，“（PK）”指示主键。

表 29. STMG_OBJECT 表

列名	数据类型	可空	描述
STMG_TIMESTAMP（PK）	TIMESTAMP	N	存储快照的时间戳记。它指示数据捕获过程开始的时间。
OBJ_ID（PK）	VARCHAR	N	在给定的存储快照时间戳记下的每个存储对象的唯一标识。
ROOT_ID	CHARACTER	N	根对象的标识。
OBJ_TYPE	INTEGER	N	与存储对象的类型相对应的整数值。请参阅第 138 页的表 25以了解阈值类型的定义。
OBJ_SCHEMA	VARCHAR	N	存储对象的模式。 当模式不适合于对象时，将使用“-”
OBJ_NAME	VARCHAR	N	存储对象的名称。
DBPG_NAME	VARCHAR	Y	对象所在的数据库分区组的名称。如果不适用，则为空。
TS_NAME	VARCHAR	Y	对象所在的表空间的名称。如果不适用，则为空。

STMG_HIST_THRESHOLD 表:

对于生成存储快照时用于分析存储对象的每个阈值，STMG_HIST_THRESHOLD 表中都包含一行。这基本上是执行快照时 SYSTOOLS.STMG_CURR_THRESHOLD 表中的内容的快照。

注： 在列中，“（PK）”指示主键。

表 30. STMG_HIST_THRESHOLD 表

列名	数据类型	可空	描述
STMG_TIMESTAMP（PK）	TIMESTAMP	N	存储快照的时间戳记。它指示数据捕获过程开始的时间。
STMG_TH_TYPE（PK）	INTEGER	N	与存储阈值类型相对应的整数值。请参阅第 139 页的表 26以了解阈值类型的定义。
OBJ_ID（PK）	VARCHAR	N	在给定的存储快照时间戳记下的每个存储对象的唯一标识。
WARNING_THRESHOLD	SMALLINT	Y	生成存储快照时为存储对象设置的警告阈值的值。
ALARM_THRESHOLD	SMALLINT	Y	生成存储快照时为存储对象设置的警报阈值的值。

STMG_DATABASE 表:

对于数据库存储快照的每个详细条目，STMG_DATABASE 表中都包含一行。

表 31. STMG_DATABASE 表

列名	数据类型	可空	描述
STMG_TIMESTAMP（PK）	TIMESTAMP	N	存储快照的时间戳记。它指示数据捕获过程何时开始。
OBJ_ID（PK）	VARCHAR	N	在给定的存储快照时间戳记下的每个存储对象的唯一标识。
COMPLETE_TIMESTAMP	TIMESTAMP	Y	对于由 OBJ_ID 列标识的数据库完成数据捕获过程时的时间戳记。
REMARKS	VARCHAR	Y	用户指定的标记 。

STMG_DBPGROUP 表:

对于数据库分区组存储快照的每个详细条目，STMG_DBPGROUP 表中都包含一行。

注: 在列中，“（PK）”指示主键。

表 32. STMG_DBPGROUP 表

列名	数据类型	可空	描述
STMG_TIMESTAMP（PK）	TIMESTAMP	N	存储快照的时间戳记。它指示数据捕获过程何时开始。
OBJ_ID（PK）	VARCHAR	N	在给定的存储快照时间戳记下的每个存储对象的唯一标识。
COMPLETE_TIMESTAMP	TIMESTAMP	Y	对于由 OBJ_ID 列标识的数据库分区组完成数据捕获过程时的时间戳记。
PARTITON_COUNT	SMALLINT	Y	包括在数据库分区组中的数据库分区数。
TARGET_LEVEL	BIGINT	Y	数据库分区组中包含的所有数据库分区的平均数据大小（以字节计）。它是均匀数据分发的目标级别。
DATA_SKEW	SMALLINT	Y	所有数据库分区中最大数据大小与 TARGET_LEVEL 偏离的百分比。在数据捕获和分析过程中，将使用此值来与在第 140 页的表 27 表中为数据库分区组设置的数据分发变形进行比较。
TOTAL_SIZE	BIGINT	Y	数据库分区组中包含的所有数据库分区的总大小（以字节计）。它是在数据库分区组中定义的所有表空间的总大小（页大小乘以页数）的总和。对于 DMS 表空间，总大小是分配的大小；对于 SMS 表空间，它是表空间当前使用的大小。
DATA_SIZE	BIGINT	Y	数据库分区组中包含的所有数据库分区的数据大小（以字节计）。它是在数据库分区组中定义的所有表空间的数据大小（页大小乘以数据页数）的总和。

表 32. STMG_DBPGROUP 表 (续)

列名	数据类型	可空	描述
PERCENT_USED	SMALLINT	Y	数据大小相对于总大小的百分比值。在数据捕获和分析过程中，将此值与空间使用阈值进行比较。在 SMS 表空间的情况下，应将表空间或者它的父数据库分区组的空间使用阈值设置为 100，以避免发生不必要的警报。
REMARKS	VARCHAR	Y	用户指定的标记 。

STMG_DBPARTITION 表:

对于数据库分区存储快照的每个详细条目，STMG_DBPARTITION 表中都包含一行。这意味着要与 STMG_DBPGROUP 表一起使用。

注: 在列中，“（PK）”指示主键。

表 33. STMG_DBPARTITION 表

列名	数据类型	可空	描述
STMG_TIMESTAMP（PK）	TIMESTAMP	N	存储快照的时间戳记。它指示数据捕获过程何时开始。
OBJ_ID（PK）	VARCHAR	N	在给定的存储快照时间戳记下的每个存储对象的唯一标识。
PARTITION_NUM（PK）	INTEGER	Y	数据库分区号。
COMPLETE_TIMESTAMP	TIMESTAMP	Y	对于由 OBJ_ID 列标识的数据库分区完成数据捕获过程时的时间戳记。
DBPG_NAME	CHARACTER	Y	数据库分区组的名称。
IN_USE	CHARACTER	Y	生成存储快照时数据库分区的状态。与 SYSCAT.DBPARTITIONGROUPDEF 中的 IN_USE 列相同。
HOST_NAME	VARCHAR	Y	数据库分区的主机名。
HOST_SYSTEM_SIZE	BIGINT	Y	不可用。
EST_DATA_SIZE	BIGINT	Y	数据库分区组作用域内的数据库分区的估计数据大小。此值是给定分区上那部分表的数据大小的总和。

STMG_TABLESPACE 表:

对于表空间存储快照的每个详细条目，STMG_TABLESPACE 表中都包含一行。

注: 在列中，“（PK）”指示主键。

表 34. STMG_TABLESPACE 表

列名	数据类型	可空	描述
STMG_TIMESTAMP（PK）	TIMESTAMP	N	存储快照的时间戳记。它指示数据捕获过程何时开始。
OBJ_ID（PK）	VARCHAR	N	在给定的存储快照时间戳记下的每个存储对象的唯一标识。

表 34. STMG_TABLESPACE 表（续）

列名	数据类型	可空	描述
COMPLETE_TIMESTAMP	TIMESTAMP	Y	对于由 OBJ_ID 列标识的表空间完成数据捕获过程时的时间戳记。
TYPE	CHARACTER	Y	如 SYSCAT.TABLESPACES 中所定义的那样。
DATATYPE	CHARACTER	Y	如 SYSCAT.TABLESPACES 中所定义的那样。
TOTAL_SIZE	BIGINT	Y	如 SYSCAT.TABLESPACES 中所定义的那样。
PERCENT_USED	SMALLINT	Y	如 SYSCAT.TABLESPACES 中所定义的那样。在数据捕获和分析过程中，使用它来与 STMG_CURR_THRESHOLD 表中的空间使用阈值进行比较。
DATA_SIZE	BIGINT	Y	DATA_PAGE * PAGE_SIZE.
DATA_PAGE	BIGINT	Y	USED_PAGES 定义在 SYSPROC.SNAPSHOT_TBS_CFG 表 UDF 中。
EXTENT_SIZE	INTEGER	Y	如 SYSCAT.TABLESPACES 中所定义的那样。
PREFETCH_SIZE	INTEGER	Y	如 SYSCAT.TABLESPACES 中所定义的那样。
OVERHEAD	DOUBLE	Y	如 SYSCAT.TABLESPACES 中所定义的那样。
TRANSFER_RATE	DOUBLE	Y	如 SYSCAT.TABLESPACES 中所定义的那样。
BUFFERPOOL_ID	INTEGER	Y	如 SYSCAT.TABLESPACES 中所定义的那样。
PAGE_SIZE	INTEGER	Y	如 SYSCAT.TABLESPACES 中所定义的那样。

STMG_CONTAINER 表:

对于容器存储快照的每个详细条目，STMG_CONTAINER 表中都包含一行。

注: 在列中，“（PK）”指示主键。

表 35. STMG_CONTAINER 表

列名	数据类型	可空	描述
STMG_TIMESTAMP（PK）	TIMESTAMP	N	存储快照的时间戳记。它指示数据捕获过程何时开始。
OBJ_ID（PK）	VARCHAR	N	在给定的存储快照时间戳记下的每个存储对象的唯一标识。
COMPLETE_TIMESTAMP	TIMESTAMP	Y	对于由 OBJ_ID 列标识的容器完成数据捕获过程时的时间戳记。
TABLESPACE_ID	INTEGER	Y	tablespace_id － 表空间标识监视元素
CONTAINER_ID	INTEGER	Y	container_id － 容器标识监视元素

表 35. STMG_CONTAINER 表 (续)

列名	数据类型	可空	描述
PARTITION_NUM	INTEGER	Y	node_number – 节点号监视元素
CONTAINER_TYPE	CHARACTER	Y	container_type – 容器类型监视元素
TOTAL_PAGES	BIGINT	Y	container_total_pages – 容器中的总页数监视元素
USABLE_PAGES	BIGINT	Y	container_usable_pages – 容器中的可用页数监视元素
ACCESSIBLE	BIGINT	Y	container_accessible – 容器的可访问性监视元素
STRIPE_SET	BIGINT	Y	container_stripe_set – 条形集监视元素
FILESYSTEM_NODENAME	BIGINT	Y	在其中定义了容器的文件系统的节点名。
FILESYSTEM_ID	BIGINT	Y	唯一文件系统标识符。
FILESYSTEM_MOUNT_POINT	VARCHAR	Y	文件系统安装点。
FILESYSTEM_TYPE_NAME	VARCHAR	Y	文件系统类型。例如，jfs、jfs2、ext2 或 ntfs。
FILESYSTEM_DEVICE_TYPE	BIGINT	Y	文件系统设备类型。
FILESYSTEM_TOTAL_SIZE	BIGINT	Y	总文件系统大小，以字节计。
FILESYSTEM_FREE_SIZE	BIGINT	Y	可用的总文件系统大小，以字节计。
REMARKS	VARCHAR	Y	用户指定的标记 。

STMG_TABLE 表:

对于指定的快照类型中包含的每个表，STMG_TABLE 表包含一行或多行。数据库快照将对数据库中的每个表插入条目。表空间快照将对指定的表空间中的每个表插入一行或多行，表快照将对快照命令中指定的表插入条目。

对于非分区的表，每个表只插入一行。对于分区表，表所在的每个表空间对应一行。例如，如果分区表散布在 5 个表空间中，则在该表的 STMG_TABLE 中有 5 行。每一行都将包含特定于表空间的信息，但有一个例外情况：与分区表的表总计相关的信息是从所有表空间获取的值的合计；在保存表总计的位置，每一行都将显示相同的值。

注: 在列中，“（PK）”指示主键。

表 36. STMG_TABLE 表

列名	数据类型	可空	描述
STMG_TIMESTAMP（PK）	TIMESTAMP	N	存储快照的时间戳记。它指示数据捕获过程何时开始。
OBJ_ID（PK）	VARCHAR	N	在给定的存储快照时间戳记下的每个存储对象的唯一标识。
COMPLETE_TIMESTAMP	TIMESTAMP	Y	对于由 OBJ_ID 列标识的表完成数据捕获过程时的时间戳记。
DBPG_NAME	VARCHAR	Y	表所在的数据库分区组的名称。
TOTAL_ROW_COUNT	BIGINT	Y	表的总行数。
AVG_ROW_COUNT	BIGINT	Y	所有部分的表的平均行计数。

表 36. STMG_TABLE 表 (续)

列名	数据类型	可空	描述
TARGET_LEVEL	BIGINT	Y	每个数据库分区中的平均数据大小（以字节计）。
DATA_SKEW	SMALLINT	Y	对于给定的表，所有部分的表中 ROW_COUNT 值偏离 TARGET_LEVEL 的最大百分比。在数据捕获和分析过程中，使用它来与 STMG_CURR_THRESHOLD 表中的数据偏差阈值进行比较。
AVG_ROW_LENGTH	BIGINT	Y	表的平均行长度。如果已经收集了此统计信息，则它将是此表中所有列的平均列长度的总和；当没有统计数据时，通过将固定列的长度与变量列的长度的百分比相加来计算此值。
COLCOUNT	INTEGER	Y	如 SYSCAT.TABLES 中所定义的寻样。
ESTIMATED_SIZE	BIGINT	Y	如 SYSCAT.TABLES 中所定义的寻样。
NPAGES	INTEGER	Y	如 SYSCAT.TABLES 中所定义的寻样。
FPAGES	INTEGER	Y	如 SYSCAT.TABLES 中所定义的寻样。
OVERFLOW	INTEGER	Y	如 SYSCAT.TABLES 中所定义的寻样。
MAIN_TBSPACE	VARCHAR	Y	如 SYSCAT.TABLES 中所定义的寻样。
INDEX_TBSPACE	VARCHAR	Y	如 SYSCAT.TABLES 中所定义的寻样。
LONG_TBSPACE	VARCHAR	Y	如 SYSCAT.TABLES 中所定义的寻样。
REMARKS	VARCHAR	Y	用户指定的标记 。
TABLE_PARTITIONED	CHAR(1)	N	指定是将表分割到一个还是多个数据分区。如果对表进行分区，则选择值“Y”，否则，选择“N”。

STMG_TBPARTITION 表:

对于表分区存储快照的每个详细条目，STMG_TBPARTITION 表中都包含一行。

注: 在列中，“（PK）”指示主键。

表 37. STMG_TBPARTITION 表

列名	数据类型	可空	描述
STMG_TIMESTAMP（PK）	TIMESTAMP	N	存储快照的时间戳记。它指示数据捕获过程何时开始。
OBJ_ID（PK）	VARCHAR	N	在给定的存储快照时间戳记下的每个存储对象的唯一标识。
PARTITION_NUM（PK）	INTEGER	N	表分区所在的数据库分区的分区号。
COMPLETE_TIMESTAMP	TIMESTAMP	Y	对于由 OBJ_ID 列标识的表分区完成数据捕获过程时的时间戳记。
DBPG_NAME	VARCHAR	Y	表所在的数据库分区组的名称。
ROWCOUNT	BIGINT	Y	此表分区中的行数。
REMARKS	VARCHAR	Y	用户指定的标记 。

STMG_INDEX 表:

对于索引存储快照的每个详细条目，STMG_INDEX 表中都包含一行。

注: 在列中，“（PK）”指示主键。

表 38. STMG_INDEX 表

列名	数据类型	可空	描述
STMG_TIMESTAMP（PK）	TIMESTAMP	N	存储快照的时间戳记。它指示数据捕获过程何时开始。
OBJ_ID（PK）	VARCHAR	N	在给定的存储快照时间戳记下的每个存储对象的唯一标识。
COMPLETE_TIMESTAMP	TIMESTAMP	Y	对于由 OBJ_ID 列标识的索引完成数据捕获过程时的时间戳记。
DBPG_NAME	VARCHAR	Y	索引所在的数据库分区组的名称。
TB_SCHEMA	VARCHAR	Y	如 SYSCAT.INDEXES 中定义的 TABNAME 一样。
TB_NAME	VARCHAR	Y	如 SYSCAT.INDEXES 中定义的 TABSCHEMA 一样。
COLCOUNT	INTEGER	Y	如 SYSCAT.INDEXES 中定义的一样。
ESTIMATED_SIZE	BIGINT	Y	如 SYSCAT.INDEXES 中定义的一样。
NLEAF	INTEGER	Y	如 SYSCAT.INDEXES 中定义的一样。
NLEVELS	SMALLINT	Y	如 SYSCAT.INDEXES 中定义的一样。
FIRSTKEYCARD	BIGINT	Y	如 SYSCAT.INDEXES 中定义的一样。
FIRST2KEYCARD	BIGINT	Y	如 SYSCAT.INDEXES 中定义的一样。
FIRST3KEYCARD	BIGINT	Y	如 SYSCAT.INDEXES 中定义的一样。
FIRST4KEYCARD	BIGINT	Y	如 SYSCAT.INDEXES 中定义的一样。
FULLKEYCARD	BIGINT	Y	如 SYSCAT.INDEXES 中定义的一样。
CLUSTERRATIO	SMALLINT	Y	如 SYSCAT.INDEXES 中所定义的那样，在数据捕获和分析过程中，使用它来与为给定索引设置的阈值进行比较。
CLUSTERFACTOR	BIGINT	Y	如 SYSCAT.INDEXES 中定义的一样。
SEQUENTIAL_PAGES	INTEGER	Y	如 SYSCAT.INDEXES 中定义的一样。
DENSITY	INTEGER	Y	如 SYSCAT.INDEXES 中定义的一样。
REMARKS	VARCHAR	Y	用户指定的标记 。

STMG_OBJ_HISTORICAL_THRESHOLDS 视图:

STMG_OBJ_HISTORICAL_THRESHOLDS 视图包含与每个捕获的快照对象相对应的一行。此视图可用来确定生成快照时对每个对象设置的阈值。还可以方便地对数据相位差、集群比率和空间用途确定哪些超过其阈值的对象。

注: 在列中，“（PK）”指示主键。

表 39. STMG_OBJ_HISTORICAL_THRESHOLDS 视图

列名	数据类型	可空	描述
STMG_TIMESTAMP（PK）	TIMESTAMP	N	存储快照的时间戳记。它指示数据捕获过程何时开始。
OBJ_ID（PK）	VARCHAR	N	在给定的存储快照时间戳记下的每个存储对象的唯一标识。
OBJ_NAME（PK）	VARCHAR	N	存储对象的名称。
OBJ_SCHEMA（PK）	VARCHAR	N	存储对象的模式。 当模式不适合于对象时，将使用“-”。
DBPG_NAME	VARCHAR	Y	对象所在的数据库分区组的名称。如果不适用，则为空。
TS_NAME	VARCHAR	Y	对象所在的表空间的名称。如果不适用，则为空。
SPACE_WARNING_THRESHOLD	SMALLINT	Y	表空间用途警告阈值。如果不适用，则为空。
SPACE_ALARM_THRESHOLD	SMALLINT	Y	表空间用途警报阈值。如果不适用，则为空。
SPACE_THRESHOLD_EXCEEDED	SMALLINT	Y	空间用途阈值超过值。如果超过的话，则为 1；否则 0。如果不适用，则为空。
SKEW_WARNING_THRESHOLD	SMALLINT	Y	数据相位差警告阈值。如果不适用，则为空。
SKEW_ALARM_THRESHOLD	SMALLINT	Y	数据相位差警报阈值。如果不适用，则为空。
SKEW_THRESHOLD_EXCEEDED	SMALLINT	Y	数据相位差阈值超过值。如果超过的话，则为 1；否则 0。如果不适用，则为空。
CLUSTER_WARNING_THRESHOLD	SMALLINT	Y	集群比率警告阈值。如果不适用，则为空。
CLUSTER_ALARM_THRESHOLD	SMALLINT	Y	集群比率警报阈值。如果不适用，则为空。
CLUSTER_THRESHOLD_EXCEEDED	SMALLINT	Y	集群比率阈值超过值。如果超过的话，则为 1；否则 0。如果不适用，则为空。

相关参考:

- 第 137 页的『“存储管理”视图』

阈值

对于存储器管理来说，阈值用于监视数据库的存储器使用情况。在“存储器管理”视图中，可以为“存储器管理”工具设置警告和警报阈值，以便与系统的实时读数进行比较。如果对象的存储器状态超出为该对象设置的安全级别或阈值，在“存储器管理”视图中就会在该对象旁显示警报标志。

创建数据库时，就为数据库对象设置了缺省阈值。其所有子代（数据库作用域内的对象）都将继承缺省阈值。然而，可以通过为任何对象提供特定值来覆盖缺省阈值。一旦为对象设置了阈值，除非另有指定，否则在其作用域下定义的所有对象都将继承其阈值设置。

“存储器管理”视图监视三种类型的阈值：空间使用情况、数据偏差和集群比率。

- “空间使用情况”量度对象使用的可用存储空间的百分比。空间使用情况是通过表空间监视的。对象的空间使用情况以存储空间总量的百分比表示，值为 0 到 100。
- “数据偏差”量度数据分布情况，这是通过量度对象与平均数据水平的偏差（以百分比计）实现的。数据偏差是通过表和数据库分区组监视的。超出数据偏差阈值后，可以使用“再分布数据”向导来在数据库分区组中各个数据库分区上平均分布数据。对象的磁盘偏差情况以百分比值表示，这个百分比值显示了对象与平均数据水平的偏差，范围为 -100 到 100。负整数表示数据水平低于平均数据水平，正数数据水平表示数据水平高于平均水平。
- “集群比率”量度表行按给定索引指定的顺序排列的程度。较高的集群比率表示数据行与索引的物理存储顺序一致。较低的集群比率表示索引和数据行以不同的物理顺序存储。集群比率以百分比表示，值为 0 到 100。

在“运行状况中心”中，量度连续值范围的运行状况指示器的条件是根据阈值定义的。阈值定义边界或区域，并且被配置为单界限的，即只有递增或递减值。有三种边界或区域：正常、警告和警报。如果运行状况指示器值处于警告区域中，就会发出警告警报。同样，如果指示器值处于警报区域中，就会生成警报。

相关参考:

- 第 137 页的『“存储管理”视图』

临时表空间设计

系统临时表空间存放数据库管理器在执行诸如排序或连接之类的操作时所需的临时数据。这些类型的操作需要额外的空间来处理结果集。数据库必须有至少一个系统临时表空间；在缺省情况下，创建数据库时会创建一个名为 TEMPSPACE1 的系统临时表空间。IBMTEMPGROUP 是此表空间的缺省数据库分区组。

用户临时表空间存放使用 DECLARE GLOBAL TEMPORARY TABLE 语句创建的表的临时数据。为了能够定义已声明临时表，至少一个用户临时表空间应该是使用相应 USE 特权创建的。USE 特权是使用 GRANT 语句授予的。用户临时表空间不是在创建数据库时缺省创建的。

建议定义一个 SMS 临时表空间，使它的页大小等于大多数常规表空间所使用的页大小。这应该适用于典型的环境和工作负载。但最好用不同的临时表空间配置和工作负载进行实验。应该考虑下列几点：

- 在大多数情况下，临时表空间是按顺序以批处理方式访问的。也就是说，插入一组行或按顺序访存一组行。因此，当需要更少的逻辑或物理页 I/O 请求来读取给定的数据量时，页大小越大，通常所获得的性能越好。当临时表的平均行大小小于页大小除以 255 的值时，情况就不会总是这样。无论是哪种页大小，任何一页上最多可有 255 行。例如，若一个查询需要 15 个字节一行的临时表，则用 4 KB 页大小的临时表空间更合适，因为 255 行可以全部包含在一个 4 KB 页中。8 KB（或更大的）页大小将在临时表的每一页上浪费至少 4 KB（或更多）字节的空间，最好不要减少需要的 I/O 请求数。

- 若在数据库中有超过一半的常规表空间使用了相同的页大小，建议您定义具有相同页大小的临时表空间。这样做的原因是这种安排可以使临时表空间与大多数或全部的常规表空间共享同一个缓冲池空间。这样可简化缓冲池的调整。
- 当使用临时表空间重组表时，该临时表空间的页大小必须与该表的页大小匹配。由于这个原因，您应确保为现有表使用的每种不同的页大小定义了临时表空间，这样才可使用临时表空间重组这些表。

  也可不用临时表空间来重组，即直接在目标表空间中重组该表。当然，这种重组要求在目标表空间中有额外的空间来完成重组过程。
- 如果由于您的工作环境的原因使您依赖于 SMS 系统临时表空间中的系统临时表，则您可能想考虑使用注册表变量 DB2_SMS_TRUNC_TMPTABLE_THRESH。以前，当不再需要系统临时表时，就会将它们截断为文件大小为零。需要新的系统临时表将对性能产生影响。如果使用此注册表变量，则允许在系统上保留非零系统临时表，以避免重复创建和截断系统临时表对性能产生的影响。
- 通常，当存在页大小不同的临时表空间时，优化器会选择缓冲池最大的临时表空间。在这种情况下，比较聪明的做法是给一个临时表空间分配一个足够大的缓冲池，而给其余临时表空间分配较小的缓冲池。这种缓冲池分配将有助于保证有效利用主存储器。例如，若目录表空间使用 4 KB 页，而其余表空间使用 8 KB 页，则最佳临时表空间配置可能是：具有一个大缓冲池的一个 8 KB 临时表空间和一个具有小一些的缓冲池的一个 4 KB 表空间。
- 一般情况下，定义具有相同页大小的多个临时表空间没有什么好处。
- 对于临时表空间而言，SMS 几乎总是比 DMS 更合适，因为：
  - 使用 DMS 与使用 SMS 相比较而言，创建临时表需要更大的开销。
  - 在 SMS 中，磁盘空间是按需要分配的；而在 DMS 中必须预分配它。预分配可能比较困难：临时表空间保存着瞬时数据，这些数据在某个时候可能会有一个非常大的峰值存储需求，但在正常情况下可能会有一个很小的平均存储需求。使用 DMS 时，必须预分配存储器的峰值需求量；而使用 SMS 时，在非高峰期间可以使用额外的磁盘空间来完成其他任务。
  - 数据库管理器尝试将临时表页保存在内存中，而不是将它们写出至磁盘。因此，DMS 的性能优点就没有那么突出。

相关概念:

- 第 107 页的『系统管理的空间』
- 第 102 页的『表空间设计』
- 第 150 页的『SMS 表空间中的临时表』

相关参考:

- 『REORG INDEXES/TABLE command』（*Command Reference*）

SMS 表空间中的临时表

缺省情况下，当不再需要 SMS 表空间中的临时表时，不会删除它们。但是，与临时表相关联的文件会被截断为长度为零。在反复使用临时表的情况下，这样做就避免了删除和重新创建临时表对性能造成的影响。

复用临时表对于这样一些用户有好处：一种用户就是其工作负载涉及到处理较小的系统（例如，Windows NT，在 Windows 中进行文件系统调用的成本相对来说比较高）的许多小型临时表的用户，另一种用户就是其磁盘存储器是分布式的并且需要网络消息才能完成文件系统操作的用户。

缺省情况下，对于保存临时表的文件，一旦不再需要它们，就会将它们截断为长度为零或截断为 DB2_SMS_TRUNC_TMPTABLE_THRESH 注册表变量中指定的扩展数据块大小。可以通过为 DB2_SMS_TRUNC_TMPTABLE_THRESH 注册表变量指定一个更大的值来设置要使用的扩展数据块的数目。如果您的工作负载反复使用大型 SMS 临时表，并且您能够保证在使用过程中保持分配空间，则应该增大与此注册表变量相关联的值。

可以通过为 DB2_SMS_TRUNC_TMPTABLE_THRESH 注册表变量指定 0 值来关闭此功能部件。如果系统具有特定的空间限制，并且您不停地遇到 SMS 临时表空间产生磁盘不足的错误，则您可能想关闭此功能部件。

与数据库的第一个连接将删除先前分配的任何文件。如果想清除现有临时表，则应该删除所有数据库连接然后再重新连接，或者取消激活数据库然后再重新激活它。如果想确保一直为临时表分配空间，则使用 ACTIVATE DATABASE 命令来启动数据库。这将避免与数据库的第一个连接产生的重复成本。

相关概念:

- 第 149 页的『临时表空间设计』

目录表空间设计

建议使用 SMS 表空间来存储数据库目录，原因如下:

- 该数据库目录由许多大小不同的表组成。当使用 DMS 表空间时，至少为每个表对象分配两个扩展数据块。根据选择的扩展数据块大小，可能产生大量已分配而未使用的空间。当使用 DMS 表空间时，应选择小的扩展数据块大小（二至四页）；否则，应使用 SMS 表空间。
- 目录表中存在大对象（LOB）列。LOB 数据不与其他数据一起保留在缓冲池中，而是每次需要时从磁盘中读取。从磁盘读取 LOB 降低了性能。因为文件系统通常有它自己的高速缓存，所以使用 SMS 表空间或在文件容器上构建的 DMS 表空间将避免可能的 I/O（若先前引用了该 LOB 的话）。

将这些因素考虑在内，SMS 表空间更适合用作目录。

另一个要考虑的因素是，将来是否需要扩大目录表空间。虽然某些平台支持扩大 SMS 容器的基本存储器，虽然可以使用重定向复原来扩大 SMS 表空间，但使用 DMS 表空间能简化添加新容器的工作。

注: 创建数据库时，定义了三个表空间，其中包括系统目录表的 SYSCATSPACE 表空间。成为所有表空间缺省值的页大小是在创建数据库时设置的。如果选择大于 4096（即 4 KB）的页大小，则目录表的页大小将限制为目录表空间的页大小为 4 KB 时的行大小。缺省数据库页大小是作为 *pagesize* 参考数据库配置参数存储的。

相关概念:

- 第 110 页的『数据库管理的空间』

- 第 107 页的『系统管理的空间』
- 第 102 页的『表空间设计』
- 『系统目录表』（*《管理指南：实施》*）

当数据在 RAID 设备上时优化表空间性能

要在将数据存放在“独立磁盘冗余阵列”（RAID）设备中时优化性能，应对使用 RAID 设备的每个表空间执行下列操作：

- 为使用 RAID 设备的表空间定义一个容器。
- 使该表空间的 EXTENTSIZE 等于或数倍于 RAID 分割区大小。
- 确保该表空间的 PREFETCHSIZE 为：
  - RAID 分割区大小与 RAID 并行设备数之积（或此积的整数倍），并且是
  - EXTENTSIZE 的一个倍数。
- 使用 DB2_PARALLEL_IO 注册表变量对表空间启用并行 I/O。

DB2_PARALLEL_IO:

当对表空间容器读写数据时，若数据库中有多个容器，则 DB2 数据库 Linux 版、UNIX 版和 Windows 版可以使用并行 I/O。但在某些情况下，对单个容器表空间启用并行 I/O 比较有利。例如，若该容器是在由多个物理磁盘组成的单个 RAID 设备上创建的，可发出并行读写调用。

要强制对只有一个容器的表空间执行并行 I/O，可使用 DB2_PARALLEL_IO 注册表变量。可将此变量设置为“*”（星号），表示每个表空间，或者可将它设置为由逗号分开的表空间标识的列表。例如：

```
    db2set DB2_PARALLEL_IO=*        {对所有表空间打开并行 I/O}
  db2set DB2_PARALLEL_IO=1,2,4,8  {对表空间 1、2、4 和表空间 8 打开并行 I/O}
```

在设置了注册表变量后，必须停止 DB2（**db2stop**），然后重新启动它（**db2start**）才能使更改生效。

由于定义了多个容器，DB2_PARALLEL_IO 还会影响表空间。如果不设置注册表变量，则 I/O 并行性等于表空间中的容器的数目。如果设置注册表变量，则 I/O 并行性等于预取大小除以扩展数据块大小的结果。如果表空间中的个别容器分布在多个物理磁盘上，则您可能想要设置注册表变量。

例如，表空间有两个容器，并且预取大小是扩展数据块大小的四倍。如果没有设置注册表变量，则对此表空间的预取请求将分为两个请求（每个请求针对两个扩展数据块）。如果预取程序可以工作，则两个预取程序可以并行处理这些请求。在设置注册表变量的情况下，对此表空间的预取请求将分为四个请求（每个请求针对一个扩展数据块），且可以使用四个预取程序并行处理这些请求。

在此示例中，如果每一个容器都有专用的单个磁盘，则对此表空间设置注册表变量将导致对这些磁盘的争用，因为两个预取程序将会同时访问两个磁盘中的每一个。但是，如果每个容器都分布在多个磁盘上，则设置注册表变量将潜在允许同时访问四个不同的磁盘。

DB2_USE_PAGE_CONTAINER_TAG:

缺省情况下，DB2 使用每个 DMS 容器（文件或设备）的第一个扩展数据块来存储容器标记。容器标记是 DB2 的容器元数据。在较早版本的 DB2 数据库系统中，使用第一页而不是第一个扩展数据块来存储容器标记，并且因此该容器中用来存储该标记的空间较少。（在 DB2 数据库系统的较早版本中，DB2_STRIPED_CONTAINERS 注册表变量用来创建带有扩展数据块大小标记的表空间。但是，因为这现在是缺省行为，所以此注册表变量不会再起作用。）

将 DB2_USE_PAGE_CONTAINER_TAG 注册表变量设置为 ON 时，创建的任何新 DMS 容器都带有一页标记，而不是一个扩展数据块标记（缺省值）。这不会对在设置注册表变量前创建的现有容器产生任何影响。

除非有非常严格的空间约束或需要与版本 8 之前的数据库的行为保持一致，否则建议不要将此注册表变量设置为 ON。

如果 RAID 设备用于表空间容器，则将此注册表变量设置为 ON 可能对 I/O 性能有负面影响。当使用 RAID 设备作为表空间容器时，建议用等于或数倍于 RAID 分割区大小的扩展数据块大小创建表空间。但是，如果将此注册表变量设置为 ON，则将使用一页容器标记并且扩展数据块将不会与 RAID 分割区对齐。因此，在 I/O 请求期间可能需要访问比最优情况更多的物理磁盘。因此强烈建议用户不要设置此注册表变量。

过程:

要创建带有一页容器标记的容器，则将此注册表变量设置为 ON，然后停止并重新启动实例:

```
db2set DB2_USE_PAGE_CONTAINER_TAG=ON
db2stop
db2start
```

要停止创建带有一页容器标记的容器，复位此注册表变量，然后停止并重新启动实例。

```
db2set DB2_USE_PAGE_CONTAINER_TAG=
db2stop
db2start
```

“控制中心”、LIST TABLESPACE CONTAINERS 命令和 GET SNAPSHOT FOR TABLESPACES 命令不显示创建的容器具有一页标记还是扩展数据块大小标记。它们使用标签“文件”或“设备”，这取决于该容器是如何创建的。要验证创建的容器是具有一页大小标记还是扩展数据块大小标记，可以使用 DB2DART 的 /DTSF 选项来转储表空间和容器信息，然后查看要了解的容器的类型字段。查询容器 API（sqlbftcq 和 sqlbtcq）可以用来创建将显示类型的简单应用程序。

相关概念:

- 第 102 页的『表空间设计』

相关参考:

- 『系统环境变量』（*《性能指南》*）

为表选择表空间时的注意事项

当确定如何将表映射至表空间时，应考虑:

- 表的分布。

  至少应该确保选择的表空间位于具有您想要的分布的数据库分区组中。

- 表中的数据量。

  若计划在一个表空间中存储许多小表，则考虑使用 SMS 充当该表空间。对于小表，DMS 表现在 I/O 和空间管理效率方面的优点就没有那么重要。一次分配一页空间的 SMS 的优点（且仅在需要时使用）却对小表更具吸引力。若一个表较大或者您需要更快地访问表中的数据，应考虑具有较小扩展数据块大小的 DMS 表空间。

  您可能希望对每个非常大的表都使用单独的表空间，而将所有的小表组合在单个表空间中。这种分隔还允许您根据表空间的使用选择适当的扩展数据块大小。

- 表数据的类型。

  例如，有的表可能包含不经常使用的历史记录数据；最终用户可能愿意接受较长的响应时间，来等待对此数据执行的查询。在这种情况下，您可能会为历史记录表使用另一个表空间，并将此表空间分配给访问速率较低的较便宜的物理设备。

  或者，您也许能够标识某些表，这些表对于使数据快速可用以及您需要快速响应时间是必不可少的。可能要将这些表置于分配给一个快速物理设备的表空间中，这样将有助于支持这些重要的数据需要。

  通过使用 DMS 表空间，还可以将表数据分发在三个不同的表空间中：一个存储索引数据；一个存储 LOB 和长型字段数据；一个存储规则表数据。这允许您选择表空间特征和支持最适合该数据的那些表空间的物理设备。例如，可能会将索引数据置于可找到的最快的设备上，这样性能可显著提高。若将一个表分布在各 DMS 表空间中，如果启用前滚恢复，应考虑一起备份和复原那些表空间。SMS 表空间不支持以此方式将数据分发在所有表空间中。

- 管理问题。

  某些管理功能可以在表空间级执行，但不能在数据库或表级执行。例如，备份表空间（而不是数据库）可以帮助您更好地利用时间和资源。它允许频繁地备份带有大量更改的表空间，同时仅偶尔地备份带有少量更改的表空间。

  可以复原数据库或表空间。若不相关的表不共享表空间，就可以选择复原数据库一个较小的部分以降低成本。

  一种好方法是将相关的表编组在一组表空间中。这些表可以通过引用约束相关，也可以通过定义的其他业务约束相关。

  若经常需要删除并重新定义特定表，则应在它自己的表空间中定义该表，因为删除一个 DMS 表空间比删除一个表更有效率。

相关概念:

- 第 131 页的『SMS 和 DMS 表空间的比较』
- 第 110 页的『数据库管理的空间』
- 第 78 页的『数据库分区组』

- 第 107 页的『系统管理的空间』

DB2 表类型

DB2 数据库 Linux 版、UNIX 版和 Windows 版提供了下列类型的表:

- 常规表，它是作为堆来实现的
- 追加方式表，它是经过优化之后主要用于 INSERT 的常规表
- 多维集群（MDC）表，它是作为实际上同时集群在多个键或维上的多个表来实现的
- 范围集群表（RCT），它是作为提供快速直接访问的数据的顺序集群来实现的
- 分区表，它是根据表的表分区键列中的值作为数据分布到多个数据分区中的表来实现的。

每种类型的表都具有自己的特征，从而使得在特定业务环境中使用它们是很有用的。对于您使用的每个表，应考虑哪种类型的表最能满足您的需要。

具有索引的常规表是“常规用途”表选项。

通过使用 ALTER TABLE 语句使常规表处于追加方式。当您需要添加新数据和检索现有数据时，例如，当您在金融环境中处理客户帐户时，追加方式表比较适合您。在追加方式表中，记录每个帐户因借款、贷款和转帐而发生的每一笔资金变化。还有一些客户想复查他们帐户上的资金变化的历史记录。

多维集群表用于数据仓储和大型数据库环境。常规表的集群索引支持数据的单维集群。MDC 表具有可以使数据集群在多个维中的优点。

范围集群表用于数据紧密集群在一个表中的一列或多列中的情况下。这些列中的最大值和最小值定义了可能值的范围。使用这些列来访问表中的记录。

与常规表相比，分区表简化了表数据转入和转出以及管理工作，并且提高了索引布置灵活性和查询处理效率。

相关概念:

- 第 159 页的『多维集群表』
- 第 155 页的『范围集群表』

相关任务:

- 『创建具体化查询表』（*《管理指南: 实施》*）
- 『创建和填充表』（*《管理指南: 实施》*）

范围集群表

范围集群表（RCT）是一种表布局方案，在此方案中，表中的每一条记录都预先确定了记录标识（RID），该标识是用来在表中查找记录的内部标识。

对于用来保存数据的每个表，应考虑最可能满足您需要的表类型。例如: 如果具有松散集群（而不是单调增大）的数据记录，则应考虑使用常规表和索引。如果有一些数据记录将在键中具有重复（不是唯一的）值，则不应使用范围集群表。如果不能在磁

盘上为您想要的范围集群表预分配固定的存储量，则不应使用这种类型的表。这些因素可帮助您确定是否具有可用作范围集群表的数据。

使用了一种算法来使记录的键值与表中特定行的位置相等。基本算法是相当简单的。采用其最基本的格式（使用单列而不是使用两列或多列来组成键），这种算法将序号映射至逻辑行号。这种算法还使用记录的键来确定逻辑页号和槽号。此过程能够特别快速地访问记录；即，访问表中的特定行。

这种算法不涉及到散列，这是因为散列不会保持键值排序。必须保持键值排序，原因是它经过一段时间之后不需要重组表数据。

表中的每个记录键都应该具有以下特征:
- 唯一
- 非空
- 整数（SMALLINT、INTEGER 或 BIGINT）
- 单调增大
- 在根据键中的每一列预先确定的一组范围内

  当创建允许键值超出所定义的范围的表时，使用 ALLOW OVERFLOW 选项。当创建键值将不超出所定义的范围的表时，使用 DISALLOW OVERFLOW 选项。在此情况下，如果将一条记录插入到由范围指示的边界外部，则会返回一条 SQL 错误消息。

分布有紧密集群（密集）的序列键的应用程序很可能是范围集群表的最佳候选应用程序。当使用这种类型的键来创建范围集群表时，该键用来生成表中的一行的逻辑位置。此过程不需要具有独立的索引。

范围集群表结构的优点包括下列因素:
- 直接访问

  通过范围集群表的键至 RID 映射功能来进行访问。
- 易于维护

  不需要在每次 INSERT、UPDATE 或 DELETE 时都要更新辅助结构（例如，B+ 树）。
- 日志记录量较少

  与大小接近的常规表和相关联的 B+ 树索引比较起来，范围集群表需要进行的日志记录量较少。
- 需要较少的缓冲池内存

  不需要额外的内存用来存储辅助结构。
- 将 B+ 树形表的属性排序

  记录的排序与通过 B+ 树形表获得的排序相同，而不需要额外级别或者 B+ 树下一个键锁定方案。使用了 RCT 之后，与常规 B+ 树形索引比较起来，代码路径长度减少了。但是，要获得这种优点，必须使用 DISALLOW OVERFLOW 来创建范围集群表，并且数据必须是密集的而不是稀疏的。
- 少一个索引

将每个键映射至磁盘上的一个位置，这意味着可以使用少一个索引（与必需的索引数比较起来）来创建表。对于范围集群表，访问表中数据的应用程序需求可能使得不需要另一个独立的索引。您仍然可以选择创建常规索引，尤其是应用程序要求这种操作的话。

索引可用来执行下列功能:

- 根据记录中的键来找到记录
- 应用启动和停止键扫描
- 垂直分布数据

  通过使用 RCT，唯一不考虑的索引属性是垂直数据分布。

当决定使用范围集群表时应考虑下列特征，这些特征使范围集群表与常规表有所不同:

- 范围集群表没有可用空间控制记录（FSCR）。
- 预分配了空间。

  预分配了表的空间并保留给表使用（即使没有填充表的记录）。创建表时，表中没有记录；但是，预分配了整个页范围。预分配是根据记录大小和要存储的最大记录数来进行的。

  – 如果在每条记录中使用了变长字段（例如，VARCHAR），则将使用该字段的最大长度，并且整个记录大小的长度是固定的。将每条记录的整个固定长度与最大记录数配合使用来确定需要的空间。
  – 这可能会导致分配不能有效利用的附加空间。
  – 如果键值稀疏，则存在未使用的空间和不良范围扫描性能。
  – 即使尚未将包含这些键值的那些行插入数据库中，范围扫描也必须访问范围内的所有可能的记录。
- 不允许修改模式。

  如果需要对范围集群表修改模式，则必须重新创建表以包括它的新模式名称和旧表中的所有数据。尤其是:

  – 不支持改变键范围。

    这是很重要的，原因是如果需要改变表的范围，则必须创建具有期望范围的新表，并且必须使用旧表中的数据来填充新表。
- 不允许重复键值。
- 不允许超出定义的范围的键值。

  这仅适用于定义为 DISALLOW OVERFLOW 的范围集群表。

  – 明确不允许存在 NULL 值。
- 范围集群索引未具体化

  在系统目录中指示了带有 RCT 键属性的索引，并且可以由优化器来选择这些索引，但是索引并不在磁盘上具体化。对于常规表，还需要为与表相关联的每个索引提供空间。对于 RCT，RCT 索引不需要空间。优化器使用引用此 RCT 索引的系统目录中的信息来确保可以选择对表的正确访问方法。
- 不允许根据范围集群表索引的定义创建主键或唯一键，因为它将是重复的。

- 范围集群表保留了原始键值排序，此功能部件将保证表中的行的集群。

除了这些注意事项之外，还存在一些不兼容性，它们要么会限制可以使用范围集群表的位置，要么会限制不使用这些表的其他实用程序。范围集群表的局限性包括:

- 不支持分区表上的范围集群表。

  如果尝试创建具有范围集群的分区表，就会返回错误消息 SQL0270 rc=87。
- 声明的全局临时表（DGTT）不受支持。

  不允许这些临时表使用范围集群属性。
- 自动摘要表（AST）不受支持。

  不允许这些表使用范围集群属性。
- load 实用程序不受支持。

  必须通过导入操作或并行插入应用程序来一次只插入一行。
- REORG TABLE 实用程序不受支持。

  将不需要重组定义为 DISALLOW OVERFLOW 的范围集群表。定义为 ALLOW OVERFLOW 的这些范围集群表仍然不允许重组此溢出区域中的数据。
- 仅在一台逻辑机器上的范围集群表。

  在具有“数据库分区功能部件”（DPF）的“企业服务器版”上，范围集群表不能存在于包含多个数据库分区的数据库中。
- 设计顾问程序将不会建议范围集群表。
- 根据定义，范围集群的表是已集群的。

  这意味着下列集群方案与范围集群表不兼容:
  - 多维集群（MDC）表
  - 集群索引
- 值和缺省压缩不受支持。
- 不支持对范围集群表进行逆向扫描。
- 不支持 IMPORT 命令使用 REPLACE 选项。
- 不支持 ALTER TABLE ... ACTIVATE NOT LOGGED INITIALLY 语句使用 WITH EMPTY TABLE 选项。

相关概念:

- 第 158 页的『范围集群表和超出范围的记录键值』
- 『范围集群表示例』（*《管理指南：实施》*）

相关参考:

- 『对本机 XML 数据存储器的限制』（*《XML 指南》*）

范围集群表和超出范围的记录键值

使用 CREATE TABLE 语句和 ALLOW OVERFLOW 选项来控制范围集群表（RCT）的行为，使它允许溢出记录。这样，可以确保立即分配表所需要的在定义范围之内的所有页。

一旦创建了记录，具有属于定义范围内的键的任何记录都将以相同方式工作，无论创建表时是否允许溢出选项。当一条具有键的记录位于定义范围之外时，情况将有所不同。在此情况下，当表允许溢出记录时，将把记录放置在动态分配的溢出区域中。当从定义范围之外添加更多记录时，将把这些记录放置在正在增加的溢出区域中。对涉及到此溢出区域的表执行的操作将需要更长的处理时间，这是因为必须将溢出区域作为操作的一部分来访问。溢出区域越大，访问溢出区域所花的时间就会越长。在延长了使用溢出区域的时间之后，可考虑通过将表中的数据导出至使用新的扩展范围定义的新范围集群表来减小溢出区域大小。

可能有些时候您不想将记录放置在范围集群表中，以使记录键值属于允许的或定义的范围之外。要使这种类型的 RCT 存在，必须在 CREATE TABLE 语句中使用 DISALLOW OVERFLOW 选项。一旦创建了这种类型的 RCT，如果记录键值属于允许的或定义的范围之外，则您必须接受错误消息。

相关参考:

- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）

范围集群表锁定

在正常的处理过程中，将锁定记录以确保在给定的任何时候都只有一个应用程序或用户有权访问记录或一组记录。对于范围集群表，不使用键锁定和下一个键锁定，而使用“离散锁定”。此方法将锁定受到或者可能会受到由应用程序或用户请求的操作影响的所有记录。获得的锁定数取决于隔离级别。

范围集群表中当前是空的但是已经预分配的那些合格行就会被锁定。这样就避免了需要进行下一个键锁定。于是，密集的范围集群表需要更少锁定。

相关概念:

- 『锁定与并行性控制』（*《性能指南》*）

多维集群表

多维集群（MDC）提供了一种极佳的方法来将多个表中的数据集群在多个维中，它具有灵活、连续并且自动完成的特点。MDC 可显著提高查询性能。此外，MDC 还可以极大地减少在插入、更新和删除操作期间执行数据维护（例如，重组）和索引维护操作的开销。MDC 主要用于数据仓储和大型数据库环境，但也可用于联机事务处理（OLTP）环境中。

相关概念:

- 『Indexes』（*SQL Reference, Volume 1*）
- 第 175 页的『MDC 表的块索引注意事项』
- 第 162 页的『块索引』
- 『MDC 表的优化策略』（*《性能指南》*）
- 『MDC 表的表和索引管理』（*《性能指南》*）
- 第 167 页的『块索引和查询性能』
- 第 172 页的『块映射』

- 第 160 页的『常规表与 MDC 表的比较』
- 第 174 页的『从 MDC 表中删除』
- 第 176 页的『设计多维集群（MDC）表』
- 第 174 页的『MDC 表的装入注意事项』
- 第 175 页的『MDC 表的记录注意事项』
- 第 170 页的『在 INSERT 操作期间自动维护集群』
- 第 183 页的『多维集群（MDC）表创建、布置和使用』
- 第 174 页的『更新 MDC 表』
- 第 164 页的『使用 MDC 表』
- 『装入数据时的多维集群注意事项』（*《数据移动指南和参考》*）

相关参考:

- 『MDC 表的表和 RID 索引扫描的锁定方式』（*《性能指南》*）
- 『MDC 表的块索引扫描的锁定』（*《性能指南》*）

常规表与 MDC 表的比较

常规表的索引是基于记录来建立的。索引的任何集群都仅限于单个维。在版本 8 之前，DB2 通用数据库仅支持通过集群索引对数据进行单维集群。通过使用集群索引，在表中插入和更新记录时，DB2 会尝试按索引的键顺序来维护各页上数据的物理顺序。集群索引极大地提高了具有包含集群索引的一个或多个键的谓词的范围查询的性能。使用好的集群索引可以提高性能，这是因为只需要访问表的一部分，并且可以执行更有效地预取。

但是，使用集群索引的数据集群有一些缺点。首先，因为随时间的推移将逐渐填满数据页上的空间，因此，将不能保证集群。插入操作将尝试把记录添加至具有相同或相似集群键值的那些页附近的页，但是，如果在理想位置没有空间，则将把它插入到表中的其他地方。因此，可能需要定期进行表重组，以便对表进行重新集群并设置具有附加可用空间的页以容纳将来的集群插入请求。

其次，只能将一个索引指定为“集群”索引，其他所有索引都将是非集群索引，这是因为只能在一维中以物理方式对数据进行集群。这种局限性与集群索引是基于记录的这样一个事实有关，因为所有索引都是在版本 8.1 之前建立的。

最后，因为对于表中的每个记录，基于记录的索引都包含一个指针，所以它们可能非常大。

集群索引

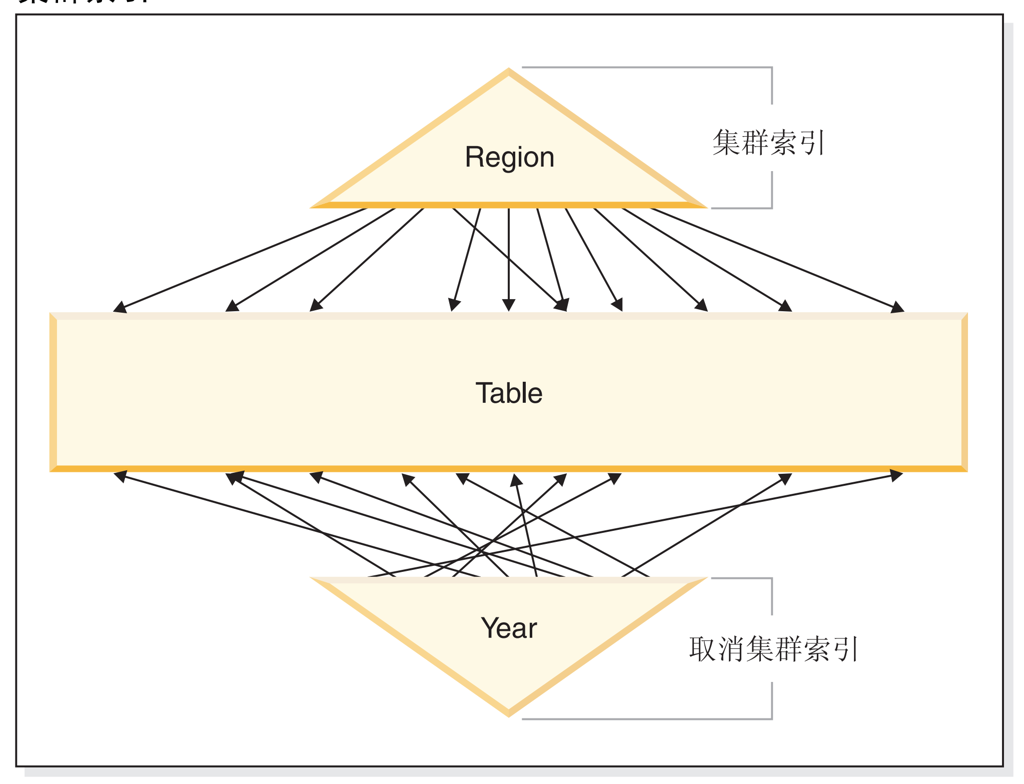


图 42. 具有集群索引的常规表

图 42 中的表上定义了两个基于记录的索引:

- “Region”上的集群索引
- “Year”上的另一个索引

“Region”索引是一个集群索引，它意味着当在索引中扫描键时，通常将在表中的同一页或邻近页上找到相应的记录。相反，“Year”索引是一个非集群索引，它意味着当在该索引中扫描键时，很可能将在表的随机页上找到相应的记录。对集群索引进行扫描将获得更好的 I/O 性能，并且在该索引的数据集群程度越高时，顺序预取就会带来更多好处。

MDC 引入了基于块的索引。“块索引”指向记录块或记录组，而不是指向单个记录。通过从物理上根据集群值将 MDC 表中的数据组织成块，然后使用块索引来访问这些块，MDC 不仅能够解决集群索引的所有缺点，还能显著地改善性能。

首先，MDC 使表能够同时在多个键或维上进行物理集群。通过 MDC，多个维或集群键也具备了单维集群的优点。在对表的一个或多个指定维具有集群的情况下，查询性能会提高。这些查询不仅是只访问包含具有正确维值的记录的那些页，还会将这些满足要求的页组成块或扩展数据块。

其次，尽管具有集群索引的表经过一段时间之后可能会变成不是集群的，但是 MDC 表还是能够对所有维自动并持续地维护和保证它的集群。因此，无需重组 MDC 表就可以复原数据的物理顺序。

最后，在 MDC 中，集群索引是基于块的。这些索引比常规的基于记录的索引要小很多，因此，占用的磁盘空间更少，并且扫描时速度会更快。

块索引

图 43 中显示的 MDC 表是以物理方式组织的，这样就可以将具有相同 “Region” 和 “Year” 值的记录组成为单独的块或扩展数据块。扩展数据块是磁盘上的一组连续页，所以将这些记录的组集群在物理上是连续的数据页上。每个表页都只属于一个块，所有块的大小都相同（即，它们的页数相同）。一个块的大小等于表空间的扩展数据块大小，以便使块边界与扩展数据块边界对齐。在这种情况下，将创建两个块索引，一个针对 “Region” 维，另一个针对 “Year” 维。这些块索引包含仅指向表中的块的指针。当对 “Region” 块索引进行扫描以找到 “Region” 等于 “East” 的所有记录时将找到两个合格的块。将在这两个块中找到 “Region” 等于 “East” 的所有记录（并且也只能找到这些记录），而且将这些记录集群在这两组连续页或扩展数据块上。同时，完全独立地扫描 “Year” 索引（1999 与 2000 之间的记录）将找到三个合格的块。对这三个块中的每个块进行数据扫描时都将返回在 1999 与 2000 之间的所有记录，并且也只能返回这些记录，并且您将发现这些记录集群在每个块的连续页上。

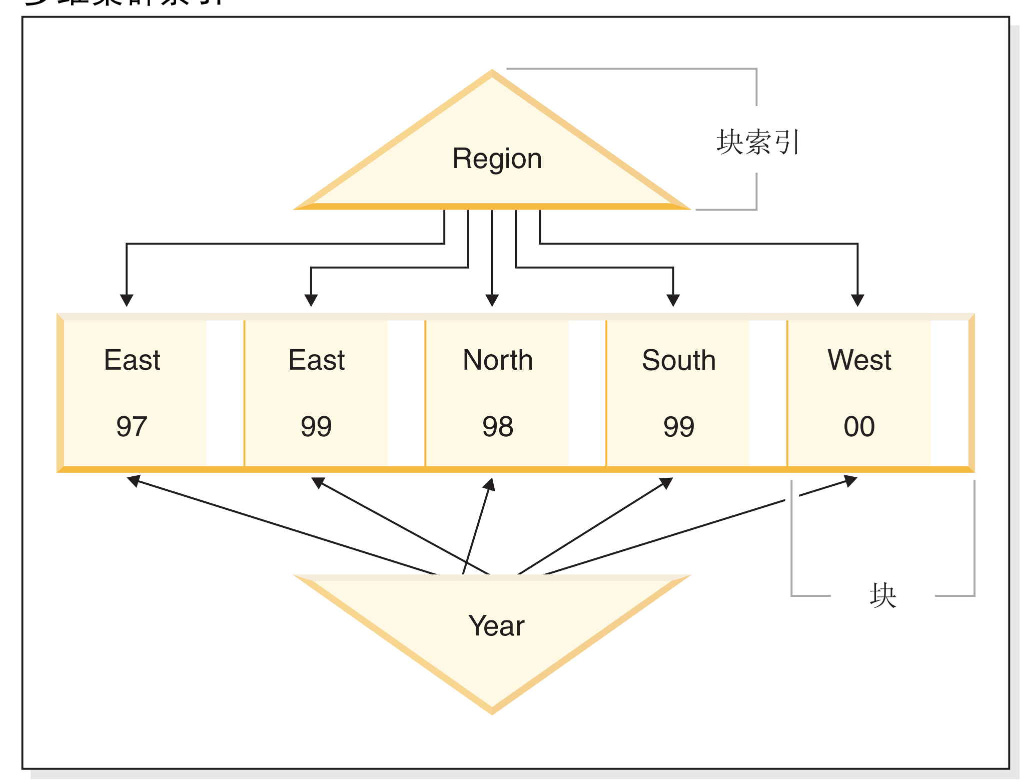


图 43. 多维集群表

除了对集群所作的这些改进之外，MDC 表还具有下列优点:

- 由于与基于记录的索引相比，由于块索引的大小是很小的，所以探测和扫描块索引要快得多
- 块索引和相应的数据组织允许更精细的 “数据库分区忽略” 或者有选择性地进行表访问
- 利用块索引的查询因为减小了索引大小、优化了对块的预取并且可以保证相应数据的集群而受益
- 对于某些查询，可以减少锁定和谓词求值

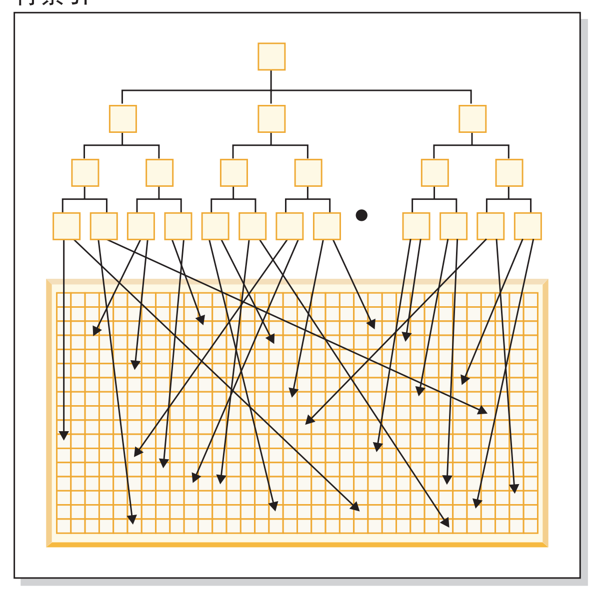
- 块索引用于日志记录和维护方面的开销很少，因为仅当将第一条记录添加至块或从块中除去最后一条记录时才需要更新它们
- 转入的数据可以复用先前转出的数据留下的连续空间。

注： 即使只是使用单个维定义的 MDC 表也可以因这些 MDC 属性而受益，并且可以是具有集群索引的常规表的可行备用。应该根据许多因素来作出此决定，这些因素包括：组成工作负载的查询以及表中数据的性质和分布。请参阅『选择维数时的注意事项』和『有关 DB2 顾问程序的 MDC 顾问程序功能部件』。

创建表时，可以将一个或多个键指定为集群数据时所采用的维。每个 MDC 维都可以包括类似于常规索引键的一列或多列。将为每个指定的维自动创建*维块索引*，并且优化器将使用它根据每个维快速并有效地访问数据。还将自动创建*组合块索引*，它包含所有维中的所有列，并且将用来在插入和更新活动期间维护数据的集群。仅当单个维尚未包含所有的维键列时，才创建组合块索引。优化器还可以选择组合块索引来有效地访问满足一部分列维或所有列维中的值的数据。

注： 此索引在处理查询期间的用途取决于它的键部分的顺序。键部分顺序是由解析器在对在 CREATE TABLE 语句的 ORGANIZE BY 子句中指定的维进行解析时遇到的列顺序来确定的。有关更多信息，请参阅『MDC 表的块索引注意事项』一节。

块索引与常规索引在结构上是相同，只不过它们指向的是块而不是记录。块索引比常规索引小的倍数为每页平均记录数乘以块大小所得到的乘积。一个块中的页数等于表空间的扩展数据块大小，它的范围是 2 到 256 页。页大小可以为 4 KB、8 KB、16 KB 或 32 KB。



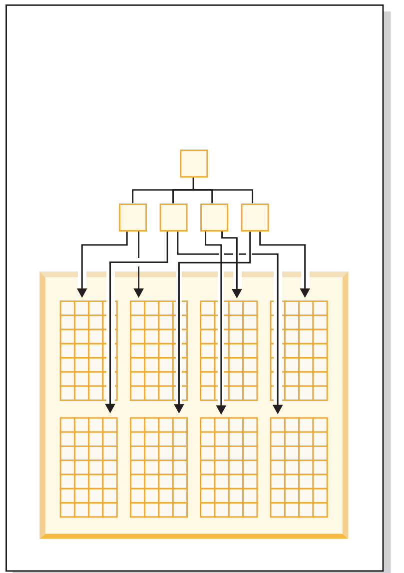


图 44. 行索引与块索引的区别

如图 44 中所示，与每一行具有单个条目相比，在块索引中，每个块都有单个索引条目。因此，块索引显著减少了磁盘的使用，并且极大地提高了数据访问速度。

在 MDC 表中，维值的每个唯一组合组成逻辑单元，它在物理上是由一个或多个页的块组成的。逻辑单元将仅使足够多的块与它相关联，以便存储具有该逻辑单元的维值的

记录。如果表中没有具有特定逻辑单元的维值的记录，则将不会为该逻辑单元分配块。包含具有特定维键值的数据的块的集合称为*片*。

相关概念:

- 第 175 页的『MDC 表的块索引注意事项』
- 第 167 页的『块索引和查询性能』
- 第 172 页的『块映射』

相关参考:

- 『对本机 XML 数据存储器的限制』（*《XML 指南》*）

使用 MDC 表

举例说明如何使用 MDC 表：我们假设一个名为“Sales”的 MDC 表，它记录有关国内零售商的销售数据。该表根据“YearAndMonth”和“Region”维进行集群。表中的记录是以块为单位存储的，块包含磁盘上足够的连续页以填充扩展数据块。在第 165 页的图 45 中，块由矩形表示，并且根据表中分配的扩展数据块的逻辑顺序进行编号。图表中的网格表示这些块的逻辑数据库分区，并且每个正方形表示一个逻辑单元。网格中的列和行表示特定维的片。例如，“Region”列中包含“South-central”值的所有记录都可以在由网格中的“South-central”列定义的片中包含的块中找到。实际上，此片中的每个块也只包含“Region”字段中具有“South-central”的记录。这样，当且仅当一个块包含在“Region”字段中具有“South-central”的记录时，该块才包含在此片或网格的列中。

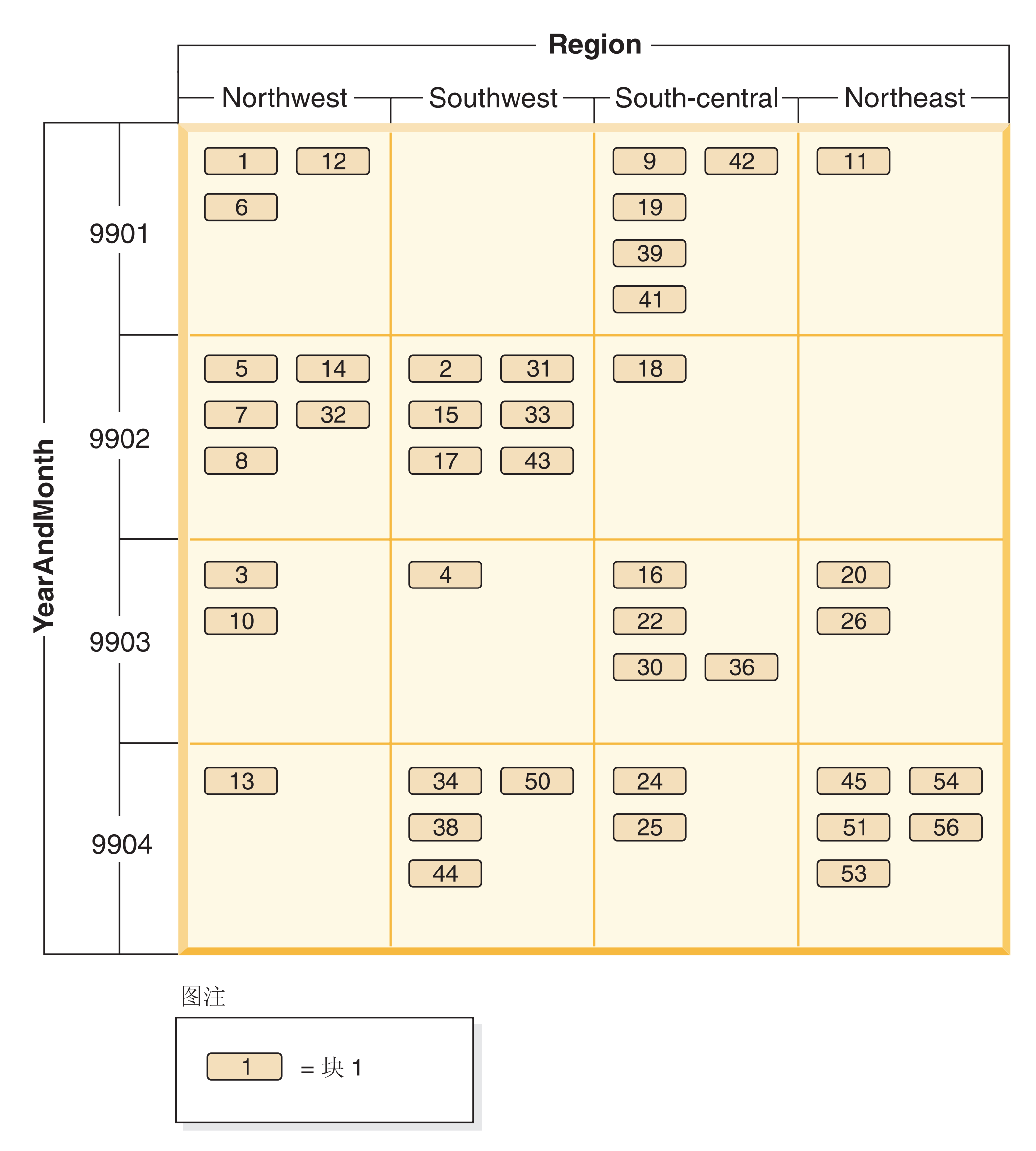


图 45. 具有 “Region” 和 “YearAndMonth” 维的称为 Sales 的多维表

为了便于确定哪些块组成片，或者哪些块包含具有特定维键值的所有记录，在创建表时将自动为每个维创建维块索引。

在第 166 页的图 46 中，一个维块索引是根据 “YearAndMonth” 维创建的，另一个维块索引是根据 “Region” 维创建的。每个维块索引的结构方式与传统 RID 索引的结构方式相同，但在叶级，这些键指向块标识（BID）而不是记录标识（RID）。RID 通过物理页号和槽号（找到记录的页上的槽）来标识表中记录的位置。BID 通过该扩展数据块的第一页的物理页号和虚槽（0）来表示块。因为块中的所有页在物理上是从该页开始连续的，并且我们知道该块的大小，所以可以使用此 BID 来找到该块中的所有记录。

片，或包含带有在维中具有特定键值的记录的页的一组块，将由该键值的 BID 列表在相关维块索引中表示。

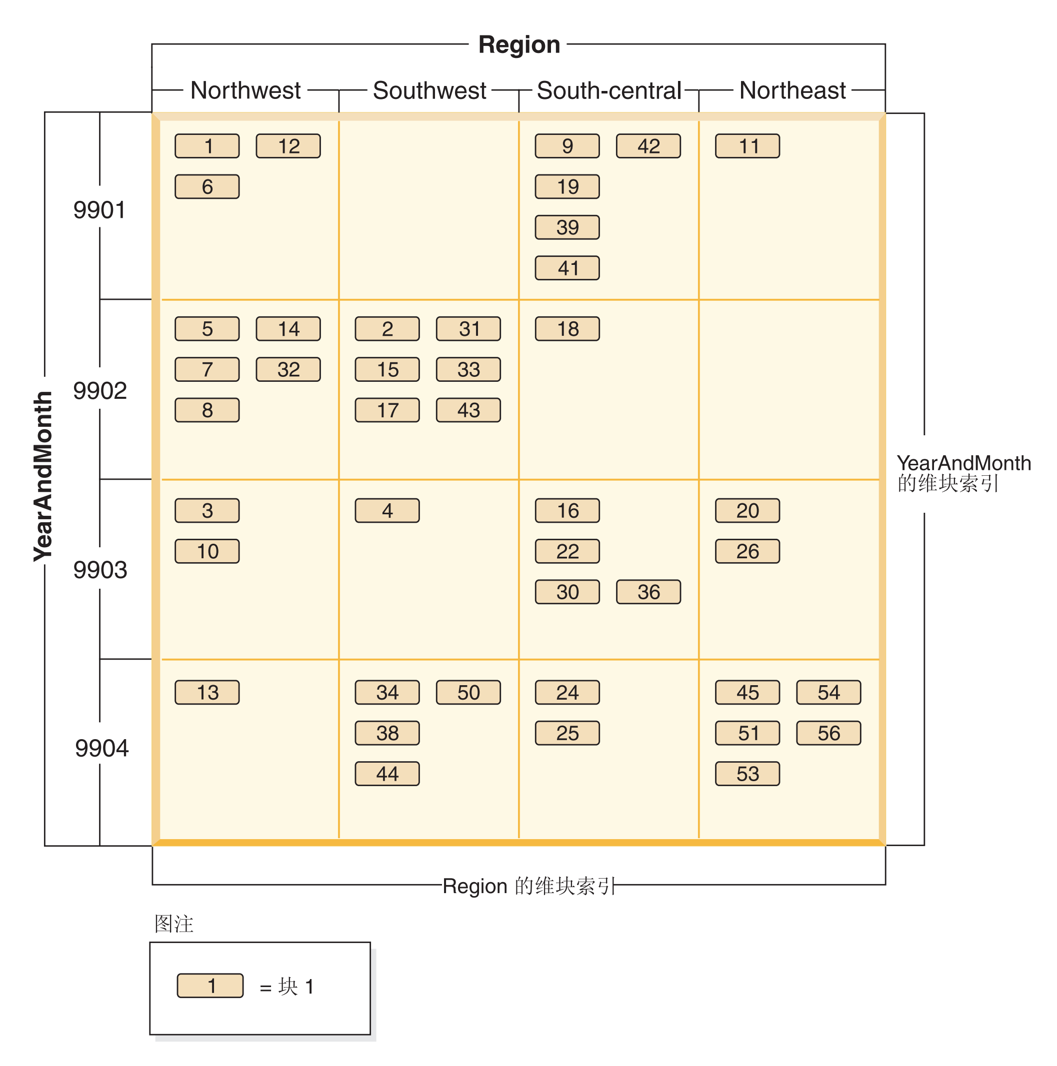


图 46. 显示维块索引且具有 “Region” 和 “YearAndMonth” 维的 Sales 表

图 47 显示 “Region” 的维块索引中的键如何出现。键由键值（即 “South-central” ）和 BID 列表组成。每个 BID 包含一个块位置。在图 47 中，列示的块号与在 Sales 表的网格中发现的 “South-central” 片相同（请参阅第 165 页的图 45）。

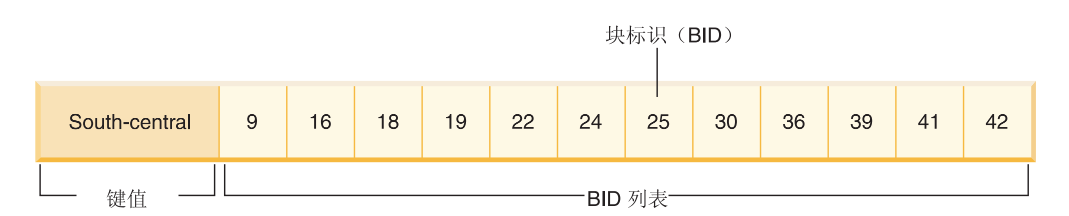


图 47. “Region” 的维块索引中的键

相似的，要找到包含 “YearAndMonth” 维为 “9902” 的所有记录的块的列表，应在 “YearAndMonth” 维块索引中查找此值，如第 167 页的图 48 中所示。

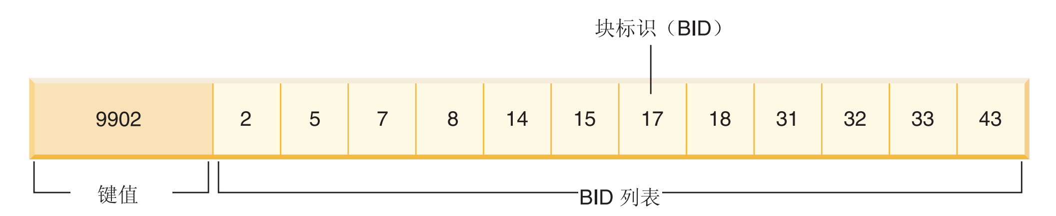


图 48. “YearAndMonth”的维块索引中的键

相关概念:

- 第 183 页的『多维集群（MDC）表创建、布置和使用』
- 第 159 页的『多维集群表』

块索引和查询性能

扫描 MDC 表的任何块索引将提供集群数据访问，这是因为每个 BID 都与保证包含具有指定维值的数据的表中的一组连续页相对应。此外，还可以通过维或片的块索引来单独访问维或片，而不会损害任何其他维或片的集群因子。这就提供了多维集群的多维性。

利用块索引访问的查询在很多方面提高了性能。第一，块索引比常规索引小很多，块索引扫描的效率很高。第二，当使用块索引时，预取数据页不依赖于顺序检测。DB2 在索引中向前查找，使用大块 I/O 将块的数据页预取到内存中并确保在表中访问数据页时扫描不会引起 I/O。第三，表中的数据集群在连续页上，这样就优化了 I/O 并将结果集定位到表的选择的部分。第四，如果在基于块的缓冲池的块大小等于扩展数据块大小的情况下使用该缓冲池，则将把 MDC 块从磁盘上的连续页预取到内存中，这样就进一步增强了集群对性能所起的作用。最后，每个块中的记录都是通过对它的数据页使用最小关系扫描来检索的，这种扫描数据的方法通常比通过基于 RID 的检索更快。

查询可以使用块索引来缩小表中具有特定维值或某一范围内的值的部分。这就提供了“数据库分区忽略”（即，块忽略）的精确格式。这样可使表具有更好的并发性，因为其他查询、装入、插入、更新和删除可访问表中的其他块，而不需要与此查询的数据集交互。

如果 Sales 表根据三个维进行集群，则还可以使用个别维块索引来查找包含满足对表的所有维的子集的查询的记录的一组块。如果表具有“YearAndMonth”、“Region”和“Product”维，则可以认为它是逻辑立方体，如第 168 页的图 49 中所示。

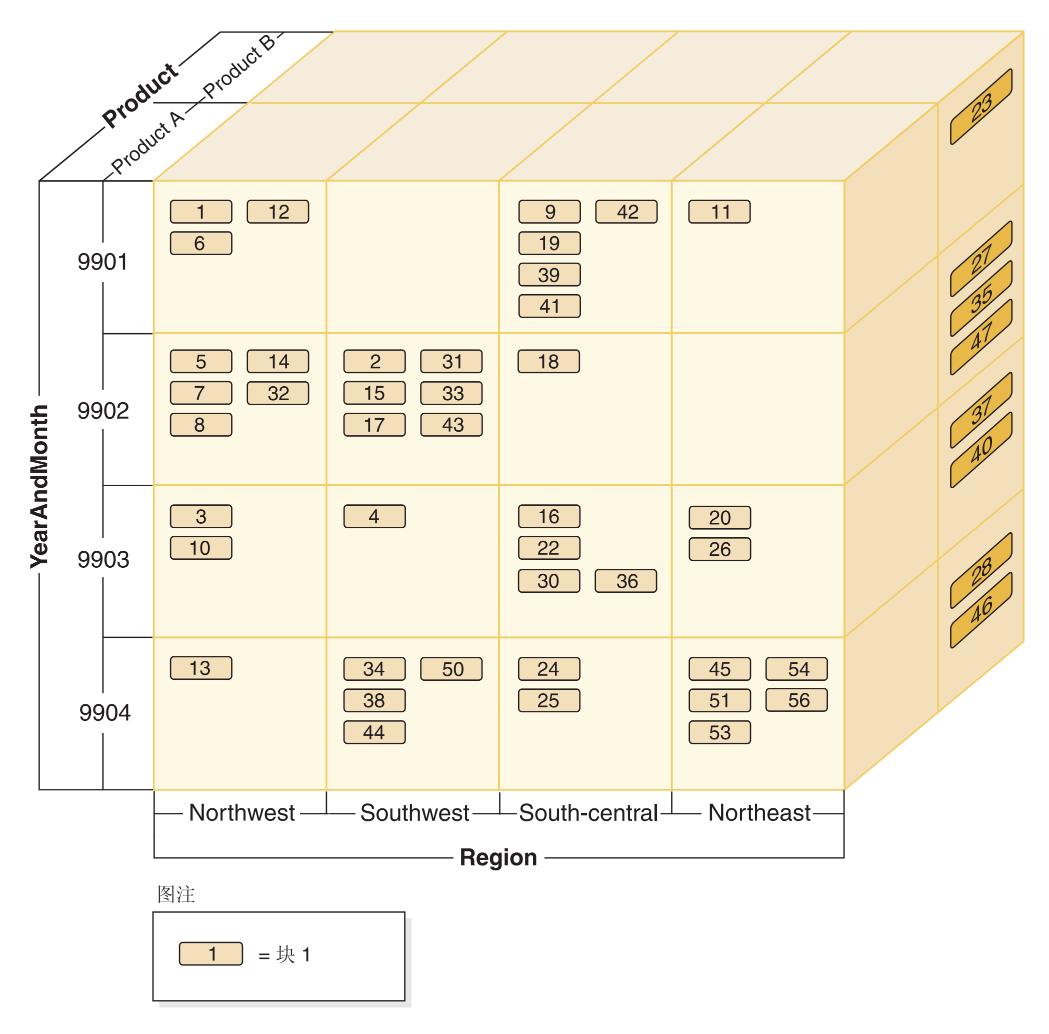


图 49. 具有 “Region” 、 “YearAndMonth” 和 “Product” 维的多维表

将对图 49 中显示的 MDC 表创建四个块索引，为“YearAndMonth”、“Region”和“Product”维每个都创建一个块索引；而另一个块索引将所有这些维列作为它的键。要检索“Product”等于“ProductA”并且“Region”等于“Northeast”的所有记录，DB2 将首先从“Product”维块索引中搜索 ProductA 键。（请参阅图 50。）然后，DB2 通过在“Region”维块索引中查找“Northeast”键来确定包含“Region”等于“Northeast”的所有记录的块。（请参阅图 51。）

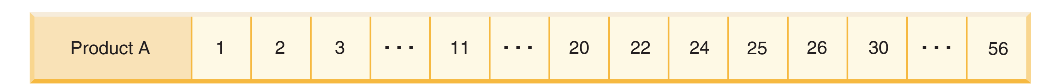


图 50. “Product” 的维块索引中的键



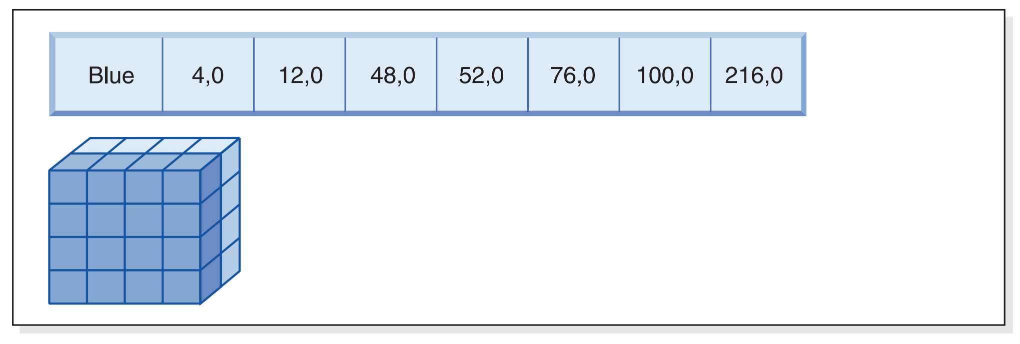
图 51. “Region” 的维块索引中的键

可以通过使用逻辑 AND 和逻辑 OR 运算符来组合块索引扫描，并且获得的要扫描的块列表还提供集群数据访问。

以上面的示例为例，要查找包含具有这两个维值的所有记录的一组块，就必须找到这两个片的交集。这是通过对两个块索引键中的 BID 列表使用逻辑 AND 操作来完成的。常用的 BID 值有 11、20、26、45、54、51、53 和 56。

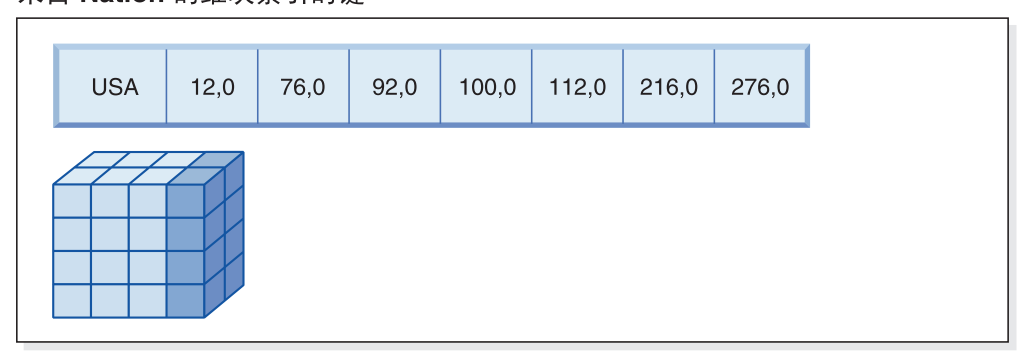
以下示例说明如何对块索引使用逻辑 OR 操作来满足具有涉及到两个维的谓词的查询。图 52 假定一个 MDC 表具有两个维："Color" 和 "Nation"。目标是检索 MDC 表中满足以下条件的所有记录："Color" 为 "blue" 或者 "Nation" 名称为 "USA"。

来自 **Colour** 的维块索引的键



+ (OR)

来自 **Nation** 的维块索引的键



=

用来扫描的块的结果块标识（**BID**）列表

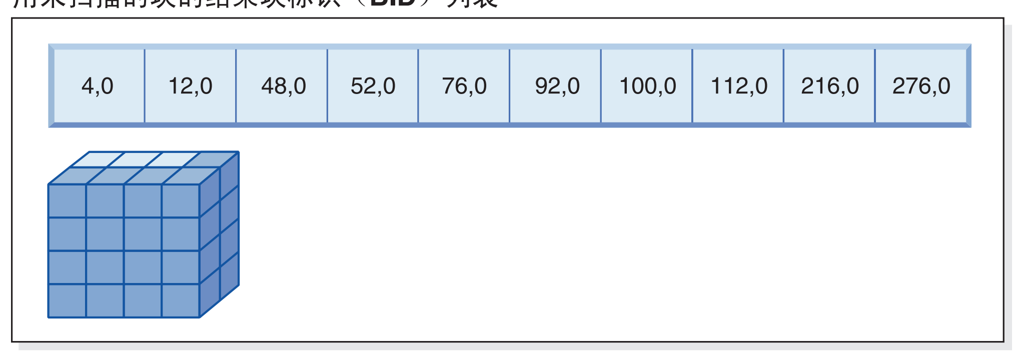


图 52. 可以如何对块索引使用逻辑 OR 操作

本图说明了如何组合两个单独的块索引扫描的结果以确定满足谓词限制的值的范围。

根据 SELECT 语句中的谓词，完成了两个单独的维块索引；一个是针对 blue 片，另一个是针对 USA 片。在内存中完成了逻辑 OR 操作，以便找到两个片的并集并确定在这两个片中找到的组合块集合（包括除去重复的块）。

一旦 DB2 具有要扫描的块列表，DB2 就可以对每个块执行最小关系扫描。可以完成块的预取，并且对于每个块，此操作将只涉及到一个 I/O，因为每个块在磁盘上作为扩展数据块存储并且可以作为一个单元读取到缓冲池中。如果需要对数据应用谓词，则只需要对块中的一个记录应用维谓词，因为将保证块中的所有记录都具有相同的维键值。如果存在其他谓词，则 DB2 只需要对块中的其余记录检查这些谓词。

MDC 表还支持基于 RID 的常规索引。可以通过使用逻辑 AND 操作或逻辑 OR 操作来将 RID 和块索引与其他索引组合在一起。块索引为优化器提供了可供选择的其他访问方案，但是您仍然可以使用传统访问方案（RID 扫描、连接、表扫描和其他方案）。在用于特定查询的所有其他可能的访问方案中，优化器将采用块索引方案，并且将选择成本最低的方案。

“DB2 设计顾问程序”可以帮助对 MDC 表建议基于 RID 的索引，或者对表建议 MDC 维。

相关概念:

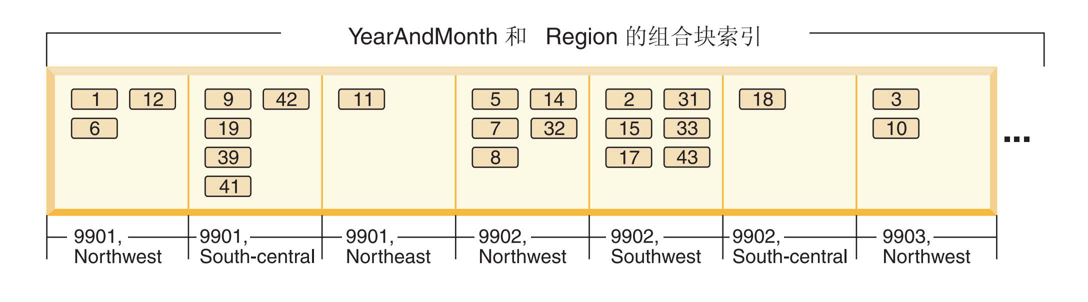
- 第 175 页的『MDC 表的块索引注意事项』
- 第 162 页的『块索引』
- 第 172 页的『块映射』

在 INSERT 操作期间自动维护集群

通过使用组合块索引确保了自动维护 MDC 表中的数据集群。它用来在 INSERT 操作期间根据表维动态管理和维护数据的物理集群。对于包含记录的表的每个逻辑单元，此组合块索引中只有一个键。因此，在 INSERT 期间使用此块索引来快速高效地确定表中是否存在逻辑单元，并且仅当表中存在逻辑单元时，才能准确地确定哪些块包含具有该单元的一组特定维值的记录。

当进行插入时:

- 将检查组合块索引，以找到与要插入的记录的维值相对应的逻辑单元。
- 如果在索引中找到了该逻辑单元的键，则它的块标识（BID）列表将提供表中具有该逻辑单元的维值的各个块的完整列表。（请参阅第 171 页的图 53。）这就限制了表的用来搜索要插入记录的空间的扩展数据块数目。
- 如果在索引中找不到该逻辑单元的键，或者如果包含这些值的扩展数据块已满，则为该逻辑单元指定新块。如果可能的话，在使用页的另一个新扩展数据块（新块）来扩展表之前，先复用该表中的空块。



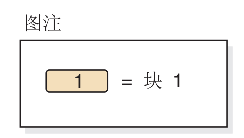


图 53. “YearAndMonth”和“Region”的组合块索引

已经保证了在包含并且仅包含具有特定维值的所有记录的一组块中将找到具有这些值的数据记录。这些块是由磁盘上的连续页组成的。因此，将按顺序访问这些记录，并且将提供集群。通过确保将记录从具有该记录的维值的单元中插入到块中来随时间推移自动维护此集群。当逻辑单元中的现有块已满时，将复用空块，或者将分配新块并将它添加至该逻辑单元的一组块。当一个块中没有数据记录时，将从块索引中除去块标识（BID）。这将使该块不再与任何逻辑单元值相关联，以便将来另一个逻辑单元可以复用它。因此，将根据需要从表中动态添加和除去单元以及与它们相关联的块索引条目，以便只容纳表中存在的数据。组合块索引用来管理这种情况，因为它将逻辑单元值映射至包含具有这些值的记录的那些块。

因为集群是按这种方式自动维护的，所以从不需要重组 MDC 表来重新集群数据。但是，仍然可以使用重组来回收空间。例如，如果单元具有许多稀疏块（数据只能满足很少的块），或者如果表具有许多指针溢出对，则重组表时就会将属于每个逻辑单元的记录压缩到需要的最小块数中，并且将除去指针溢出对。

以下示例说明了可以如何将组合块索引用于查询处理。如果想要查找 Sales 表中“Region”为“Northwest”且“YearAndMonth”为“9903”的所有记录，则 DB2 将在组合块索引中查找键值 9903，Northwest，如图 54 中所示。键由键值（即“9903，Northwest”）和 BID 列表组成。可以看到列示的 BID 仅有 3 和 10，并且实际上 Sales 表中只有两个块包含具有这两个特定值的记录。

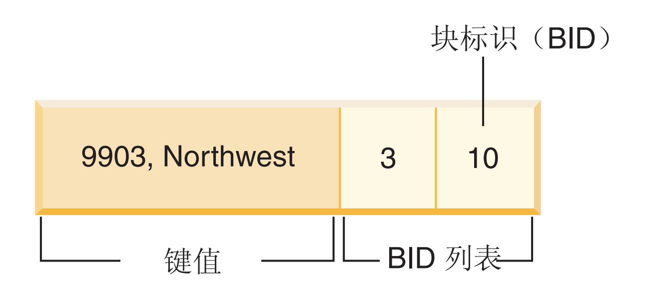


图 54. “YearAndMonth”和“Region”的组合块索引中的键

为了说明在插入期间如何使用组合块索引，我们以插入具有维值 9903 和 Northwest 的另一条记录为例。DB2 将在组合块索引中查找此键值，并查找第 3 个和第 10 个块的 BID。这些块包含具有这些维键值的仅有全部记录。如果具有可用空间，则 DB2 将把新记录插入到其中一个块中。如果在这些块中的所有页上都没有空间，则 DB2 将为表分

配新块或使用表中先前清空的块。注意，在此示例中，表当前没有使用第 48 个块。DB2 将记录插入到该块中，并通过将该块的 BID 添加至组合块索引和每个维块索引来使此块与当前逻辑单元相关联。请参阅图 55 以获取有关添加第 48 个块之后的维块索引的键的说明。

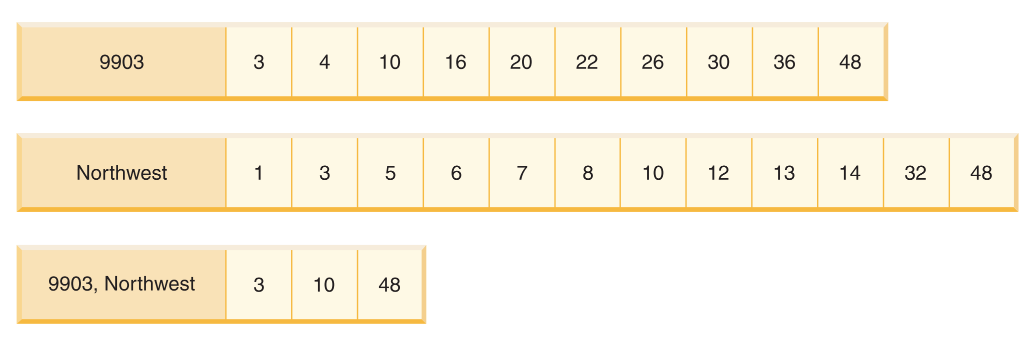


图 55. 添加第 48 个块之后维块索引中的键

相关概念:

- 第 172 页的『块映射』
- 第 162 页的『块索引』

块映射

当块被清空时，它就会通过从块索引中除去它的 BID 来不再与它的当前逻辑单元值相关联。然后，该块可被另一个逻辑单元复用。这就不需要使用新块来扩展表。当需要新块时，需要快速找到先前被清空的块，而不必搜索表来找到它们。

块映射是用来便于找到 MDC 表中的空块的新结构。块映射是作为单独的对象存储的:

- 在 SMS 中，作为单独的 .BKM 文件
- 在 DMS 中，作为对象表中的新对象描述符。

块映射是包含表的每个块的条目的一个数组。每个条目组成块的一组状态位。状态位包括:

- 正在使用。块被指定给逻辑单元。
- 装入。最近装入了块；但是通过扫描还看不见。
- 约束。最近装入了块；但是仍然需要完成约束检查。
- 刷新。最近装入了块；但是仍然需要刷新具体化查询视图。

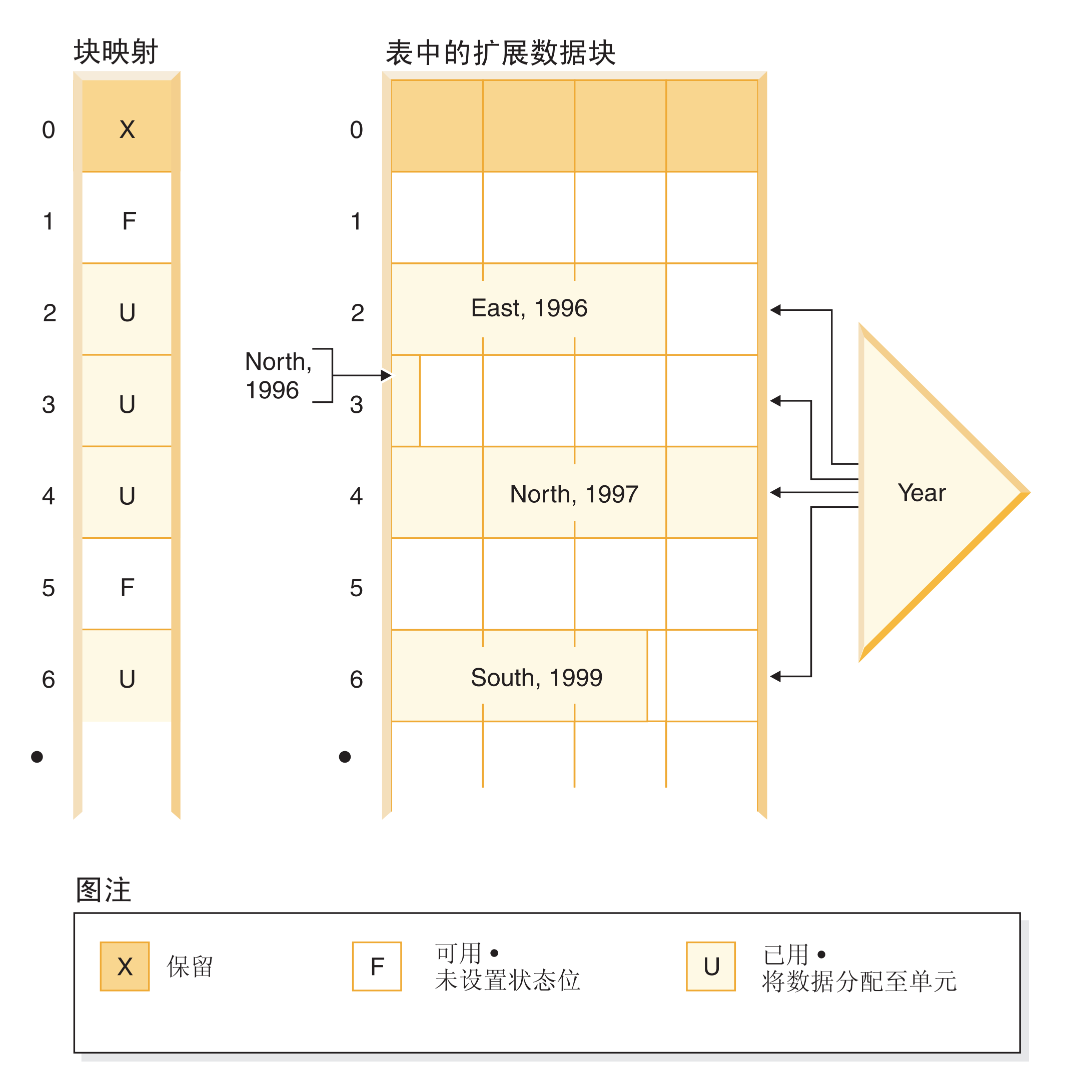


图 56. 块映射的工作方式

在图 56 的左边显示具有表中每个块的不同条目的块映射数组。右边显示了正在如何使用表的每个扩展数据块：某些扩展数据块是可用的，而大多数扩展数据块正处于使用状态，并且仅在块映射中标记为正在使用的块中找到了记录。为了简单起见，在图中只显示两个维块索引的其中一个。

注:

1. 块索引中存在仅指向块映射中标记为 IN USE 的那些块的指针。
2. 第一个块是保留块。此块包含表的系统记录。

通过扫描块映射以找到可用块（即，未设置任何位的那些块）来很容易找到可用块以便在单元中使用。

表扫描还使用块映射来仅访问当前包含数据的扩展数据块。根本不需要在表扫描中包括任何未在使用的扩展数据块。为了便于说明，此示例中的表扫描（图 56）将从表中的第三个扩展数据块（扩展数据块 2）开始，跳过第一个保留的扩展数据块和紧接着的空扩展数据块，扫描表中的第 2、3 和 4 个块，跳过下一个扩展数据块（不使用该扩展数据块的任何数据页），然后从那里继续扫描。

相关概念:

- 第 175 页的『MDC 表的块索引注意事项』
- 第 162 页的『块索引』
- 第 167 页的『块索引和查询性能』

从 MDC 表中删除

当在 MDC 表中删除记录时，如果它不是块中的最后一条记录，则 DB2 数据库系统只删除该记录并从在该表上定义的任何基于记录的索引中除去它的 RID。但是，当删除操作除去块中的最后一条记录时，DB2 通过更改它的 IN_USE 状态位并从所有块索引中除去该块的 BID 来释放该块。同样，如果还存在基于记录的索引，则也会除去它们的 RID。

注: 因此，仅当块被彻底清空时，才需要对整个块除去一次块索引条目，而不是对基于记录的索引中已删除的每一行都除去一次。

相关概念:

- 第 159 页的『多维集群表』

更新 MDC 表

在 MDC 表中，更新非维值是在适当位置完成的，正如对常规表更新非维值一样。如果更新影响了变长列并且记录不再适合页，则将查找另一个具有足够空间的页。在同一个块中开始搜索此新页。如果该块中没有空间，则使用插入新记录的算法来查找逻辑单元中具有足够空间的页。除非在单元中找不到空间并且需要将新块添加至该单元，否则不需要更新块索引。

更新维值被视作删除当前记录之后插入已更改的记录，因为该记录将更改它所属于的逻辑单元。如果删除当前记录导致块被清空，则需要更新块索引。类似地，如果插入新记录要求将它插入到新块中，则需要更新块索引。

仅当将第一条记录插入到块中或者从块中删除最后一条记录时才需要更新块索引。因此，与块索引进行维护和日志记录相关联的索引开销小于与常规索引相关联的索引开销。对于在其他情况下将为常规索引的每个块索引，极大地减少了维护和日志记录开销。

MDC 表被视作任何现有表；即，可以对这些表定义触发器、引用完整性、视图和具体化查询表。

相关概念:

- 第 159 页的『多维集群表』

MDC 表的装入注意事项

如果定期将数据装入到数据仓库中，则使用 MDC 表会很有帮助。在 MDC 表中，装入会先复用表中先前清空的块，然后才扩展该表并添加新块以便装入余下的数据。在删除了一组数据之后（例如，一个月的数据），可以使用 Load 实用程序来装入下一个月的数据，并且它可以复用在（已落实）删除之后已经清空的块。

当将数据装入 MDC 表中时，可以对输入数据排序，也可以不对它进行排序。如果不进行排序，则考虑执行下列操作:

- 增大 *util_heap* 配置参数。

  增大实用程序堆大小将影响数据库中的所有装入操作（以及备份和复原操作）。
- 增大使用 LOAD 命令的 DATA BUFFER 子句给定的值。

  增大此值将影响单个装入请求。实用程序堆大小必须足够大，以满足多个并发装入请求的可能性。
- 确保用于缓冲池的页大小与临时表空间的最大页大小相同。

装入从块边界开始，所以最好将它用于属于新单元的数据或者用于表的初始填充。

相关概念:

- 第 159 页的『多维集群表』

MDC 表的记录注意事项

如果 RID 索引先前索引或另外索引的列现在是维并且因此由块索引来索引，则索引维护和日志记录将显著地减少。仅当删除了整个块中的最后一条记录时，DB2 才需要从块索引中除去 BID 并记录此索引操作。同样，仅当将记录插入到新块中（如果它是逻辑单元的第一条记录或者是插入到当前已满的块的逻辑单元中）时，DB2 才需要将 BID 插入到块索引中并记录该操作。因为块可以具有 2 到 256 页记录，所以此块索引维护和日志记录相对较小。仍将记录表和 RID 索引的插入和删除操作。

相关概念:

- 第 159 页的『多维集群表』

MDC 表的块索引注意事项

当定义 MDC 表的维时，将创建维块索引。此外，当定义了多个维时，还可能会创建组合块索引。但是，如果为 MDC 表只定义了一个维，则 DB2 将只创建一个块索引，它将同时充当维块索引和组合块索引。相似的，如果创建的 MDC 表具有针对列 A 和针对（列 A 和列 B）的维，DB2 将对列 A 创建维块索引，并对列 A 和列 B 创建维块索引。因为组合块索引是表中的所有维的块索引，所以列 A 和列 B 的维块索引也将充当组合块索引。

组合块索引也用于查询处理，以访问在表中具有特定维值的数据。注意，组合块索引中的键部分的顺序可能会影响它对于查询处理的使用或适用性。它的键部分的顺序由创建 MDC 表时使用的整个 ORGANIZE BY DIMENSIONS 子句中的列的顺序确定。例如，如果表是使用下列语句创建的:

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c4, (c3,c1), c2)
```

则将对列 (c1,c4,c3,c2) 创建组合块索引。注意，尽管 c1 在维子句中指定了两次，但是它仅在作为组合块索引的键部分时使用了一次，并且是以第一次发现它的顺序使用

的。组合块索引中的键部分的顺序对于插入处理没有影响，但对于查询处理可能是有影响的。因此，如果更希望组合块索引具有列顺序 (c1,c2,c3,c4)，则应使用以下语句来创建表:

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
ORGANIZE BY DIMENSIONS (c1, c2, (c3,c1), c4)
```

相关概念:

- 第 162 页的『块索引』
- 第 167 页的『块索引和查询性能』
- 第 172 页的『块映射』

设计多维集群（MDC）表

一旦您决定使用多维集群表，您选择的维将不仅取决于将使用表并且受益于块级别集群的查询类型，更重要的是取决于实际数据的数量和分布情况。使用相关的概念链接可以看到设计 MDC 表的各个方面以及有关选择适当的维和块大小的一些指导。

将受益于 MDC 的查询:

选择表的集群维时首先应注意确定哪些查询将受益于块级别的集群。通常，当根据组成将对数据完成的工作的查询来选择维时有一些候选维。这些候选维的等级是很重要的。在等同查询或范围谓词查询中涉及到的列（特别是具有较小基数的列）将最受益于集群维，因此应将它们看作集群维的候选维。您还要考虑为涉及与维表进行星型连接的 MDC 事实表中的外键创建维。记住，对多个维进行自动持续的集群以及扩展数据块或块级别的集群会提高性能。

有许多种查询都可以利用多维集群。以下是这样一些查询的示例。在这些示例的某些示例中，假定存在一个 MDC 表 t1，具有 c1、c2 和 c3 维。在其他示例中，假定存在一个 MDC 表 mdctable，具有 color 和 nation 维。

示例 1:

```
SELECT .... FROM t1 WHERE c3 < 5000
```

此查询涉及到单个维的范围谓词，因此可以在内部重写以使用 c3 的维块索引来访问该表。将扫描索引以查找键值小于 5000 的块标识（BID），并且对块的结果集应用最小关系扫描以检索实际记录。

示例 2:

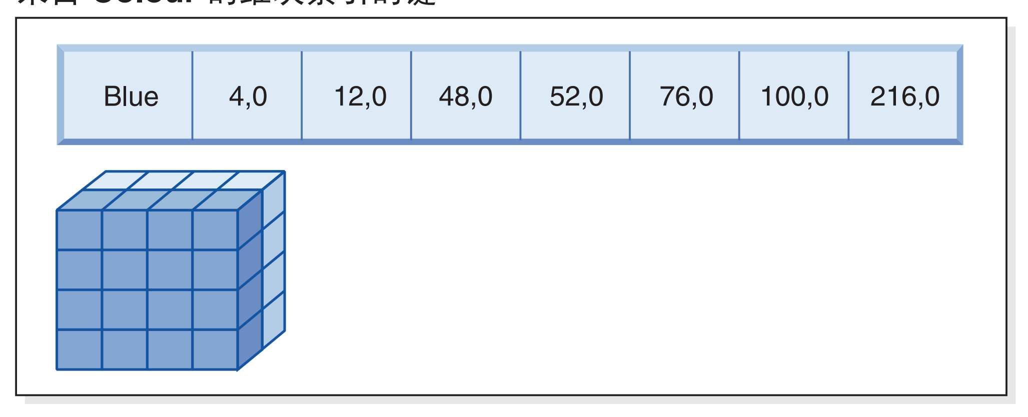
```
SELECT .... FROM t1 WHERE c2 IN (1,2037)
```

此查询涉及到单个维的 IN 谓词，并且可以触发基于块索引的扫描。可以内部重写此查询以使用 c2 的维块索引来访问该表。将扫描索引以查找具有键值 1 和 2037 的 BID，并且对块的结果集应用最小关系扫描以检索实际记录。

示例 3:

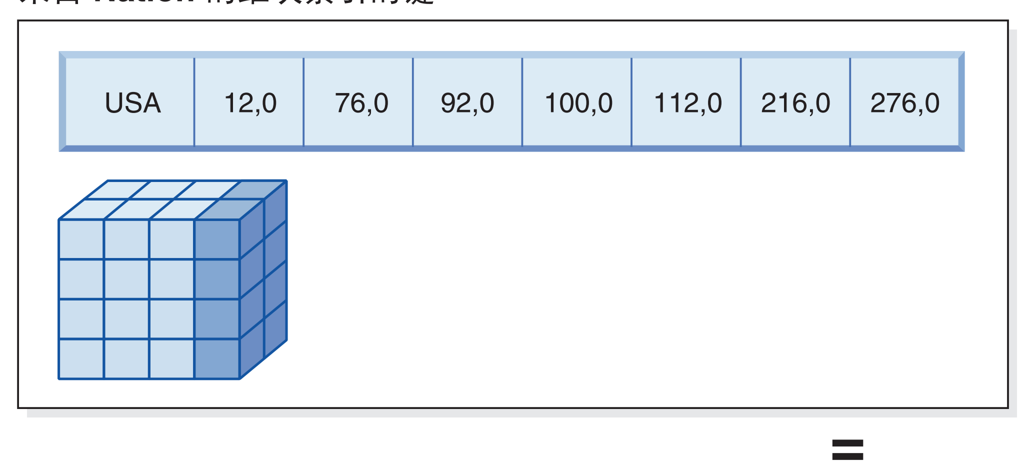
```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' AND NATION='USA'
```

来自 **Colour** 的维块索引的键



+ (AND)

来自 **Nation** 的维块索引的键



=

用来扫描的块的结果块标识（**BID**）列表

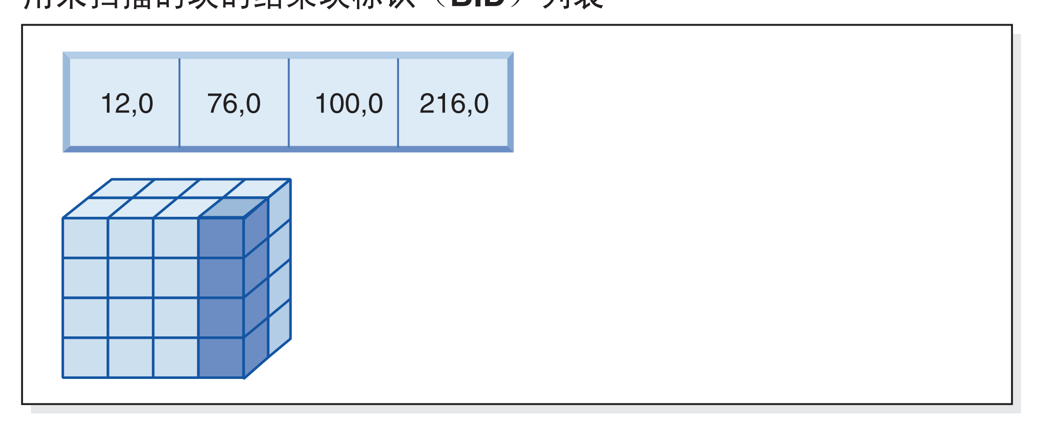


图 57. 对两个块索引使用逻辑 AND 操作的查询请求。

要执行此查询请求，完成下列步骤（显示在图 57 中）：

- 完成维块索引查找: 对 Blue 片执行一次，对 USA 片执行一次。
- 执行块逻辑 AND 操作以确定两个片的交集。即，逻辑 AND 操作只确定在这两个片中都可以找到的那些块。
- 对表中获得的块执行最小关系扫描。

示例 4:

```
SELECT ... FROM t1
   WHERE c2 > 100 AND c1 = '16/03/1999' AND c3 > 1000 AND c3 < 5000
```

此查询涉及到 c2 和 c3 的范围谓词和 c1 的等价谓词，并且还要执行逻辑 AND 操作。可以内部重写此查询以访问每个维块索引上的表:

- 扫描 c2 块索引以查找具有大于 100 的键值的 BID
- 扫描 c3 块索引以查找键值在 1000 到 5000 之间的 BID
- 扫描 c1 块索引以查找键值为“16/03/1999”的 BID。

然后，对从每个块扫描获得的 BID 执行逻辑 AND 操作以查找它们的交集，并对块的结果集应用最小关系扫描以查找实际记录。

示例 5:

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' OR NATION='USA'
```

要执行此查询请求，完成下列步骤:

- 完成维块索引查找：对每个片都执行一次。
- 执行逻辑 OR 操作以查找两个片的并集。
- 对表中获得的块执行最小关系扫描。

示例 6:

```
SELECT .... FROM t1 WHERE c1 < 5000 OR c2 IN (1,2,3)
```

此查询涉及到 c1 维的范围谓词和 c2 维的 IN 谓词，以及逻辑 OR 操作。可以内部重写此查询以访问维块索引 c1 和 c2 上的表。扫描 c1 维块索引以查找小于 5000 的值，并对 c2 维块索引执行另一个扫描以查找值 1、2 和 3。对从每个块索引扫描获得的 BID 执行逻辑 OR 操作，然后对块的结果集应用最小关系扫描以查找实际记录。

示例 7:

```
SELECT .... FROM t1 WHERE c1 = 15 AND c4 < 12
```

此查询涉及到 c1 维的等价谓词和不是维的一列的另一个范围谓词以及逻辑 AND 操作。可以内部重写该查询以访问 c1 的维块索引，以获取 c1 值为 15 的表的片的块列表。如果 c4 具有 RID 索引，则可以进行索引扫描以检索 c4 小于 12 的记录的 RID，然后可以对获得的块列表和此记录列表执行逻辑 AND 操作。此交集将除去 c1 为 15 的块中不存在的 RID，而仅从表中检索合格块中存在的列示的 RID。

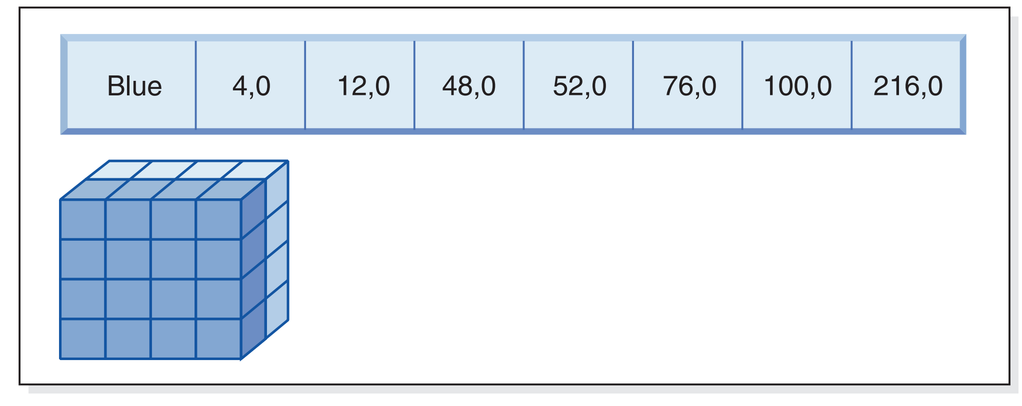
如果 c4 没有 RID 索引，则可以扫描块索引以查找合格块的列表，并且在每个块的最小关系扫描期间，可以对发现的每个记录应用谓词 c4 < 12。

示例 8:

假定这样一个方案，存在 color、year 和 nation 维以及部件号的行标识（RID）索引，则可以进行以下查询。

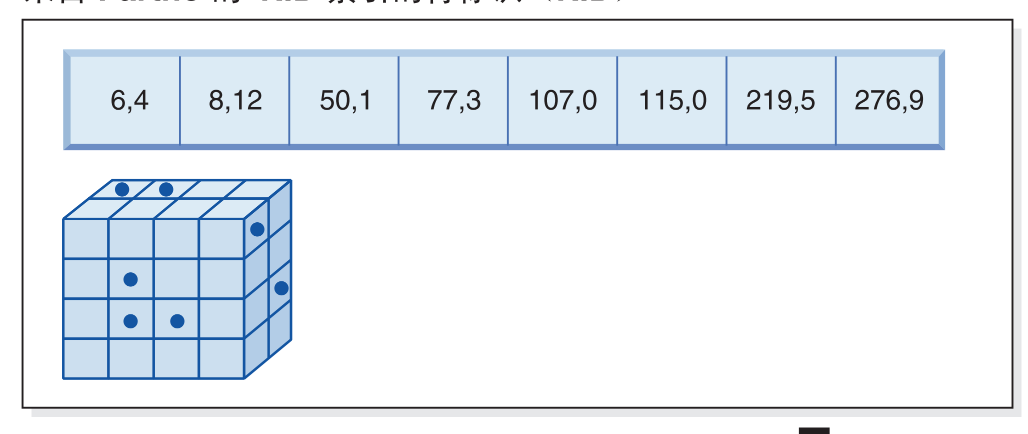
```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' AND PARTNO < 1000
```

来自 **Colour** 的维块索引的键



+ (AND)

来自 **Partno** 的 **RID** 索引的行标识（**RID**）



=

用来访存的结果行标识

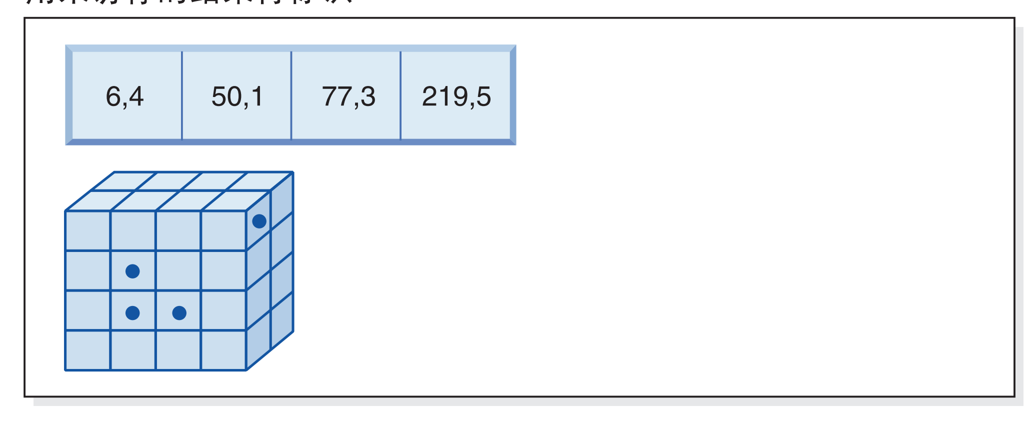


图 58. 对块索引和行标识（RID）索引使用逻辑 AND 操作的查询请求

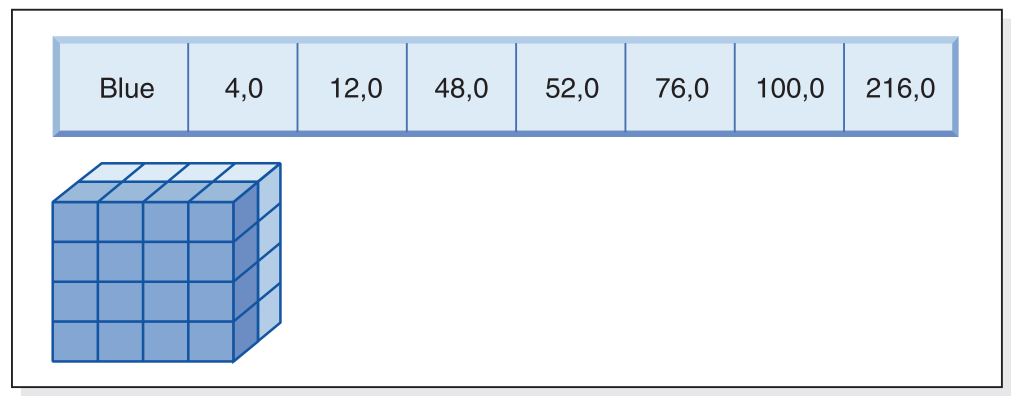
要执行此查询请求，完成下列步骤（显示在图 58 中）：

- 完成维块索引查找和 RID 索引查找。
- 对块和 RID 使用逻辑 AND 操作来确定片与满足谓词条件的那些行的交集。
- 获得的结果仅为还属于满足条件的块的那些 RID。

示例 9:

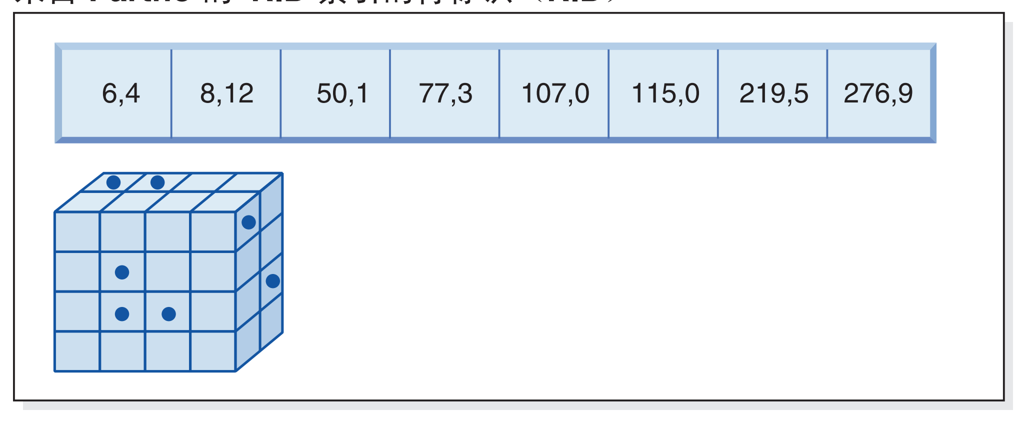
```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' OR PARTNO < 1000
```

来自 **Colour** 的维块索引的键



+ (OR)

来自 **Partno** 的 **RID** 索引的行标识（**RID**）



=

用来访存的结果块和 **RID**

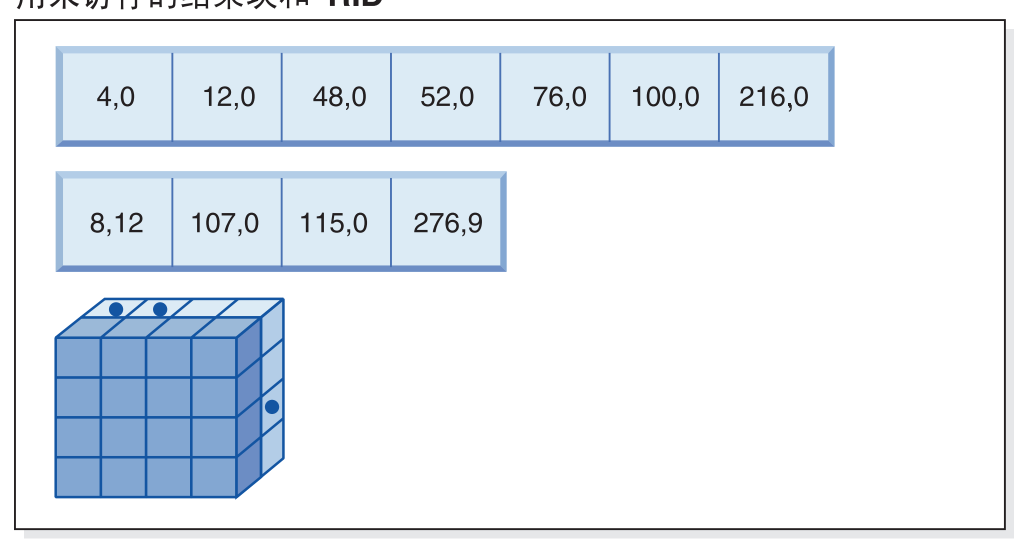


图 59. 使用逻辑 OR 操作的块索引和行标识的工作方式

要执行此查询请求，完成下列步骤（显示在第 180 页的图 59 中）：

- 完成维块索引查找和 RID 索引查找。
- 对块和 RID 使用逻辑 OR 操作来确定片与满足谓词条件的那些行的并集。
- 获得的结果是满足条件的块中的所有行以及满足谓词条件但在满足条件的块外部的其他 RID。对每个块执行最小关系扫描以检索它们的记录并逐个检索这些块之外的其他记录。

示例 10:

```
SELECT ... FROM t1 WHERE c1 < 5 OR c4 = 100
```

此查询涉及到 c1 维的范围谓词和非维列 c4 的等价谓词，以及逻辑 OR 操作。如果 c4 列上具有 RID 索引，则可以内部重写此查询以使用 c1 上的维块索引和 c4 上的 RID 索引来执行 OR 操作。如果 c4 上没有索引，则可以选择表扫描，因为必须检查所有记录。逻辑 OR 操作将对 c1 使用块索引扫描以找到小于 4 的值，以及对 c4 使用 RID 索引扫描以找到值 100。对每个满足条件的块执行最小关系扫描，因为这些块中的所有记录都满足条件，还要检索这些块外部的记录的任何其他 RID。

示例 11:

```
SELECT .... FROM t1,d1,d2,d3
   WHERE t1.c1 = d1.c1 and d1.region = 'NY'
      AND t2.c2 = d2.c3 and d2.year='1994'
      AND t3.c3 = d3.c3 and d3.product='basketball'
```

此查询涉及星型连接。在此示例中，t1 是事实表并且它具有外键 c1、c2 和 c3（与主键 d1、d2 和 d3 相对应）以及维表。维表不一定是 MDC 表。Region、year 和 product 是可以使用常规或块索引建立索引的各自维表的列（如果维表是 MDC 表的话）。当根据 c1、c2 和 c3 值访问事实表时，可以对这些列的维块索引进行块索引扫描，然后使用获得的 BID 来执行逻辑 AND 操作。当存在块列表时，可以对每个块进行最小关系扫描以获取记录。

单元的密度:

选择适当的维和扩展数据块大小对于 MDC 设计来说**非常**重要。这些因素将确定表的期望单元密度。它们之所以重要是因为对每个现有单元都分配扩展数据块，无论该单元中的记录数是多少。正确的选择将利用基于块的索引和多维集群，从而获得更好的性能。目标是获得填充密度很高的块，以便从多维集群中获得最多益处并且最佳地利用空间。

因此，设计多维表时的非常重要的注意事项就是表中的期望单元密度（基于现有的和预期的数据）。可以根据查询性能选择一组维，它导致表中的潜在单元数很大（基于每个维的可能值的数目）。表中的可能单元的数目等于每个维的基数的笛卡尔乘积。例如，如果根据 Day、Region 和 Product 维来集群表且涵盖五年的数据，则表中可能具有 `1821 days * 12 regions * 5 products = 109260` 个不同的单元。任何仅包含几个记录的单元仍需要分配给它的整块的页，以存储该单元的记录。如果块大小较大，则此表结果可能会比它实际需要的大得多。

要获得最佳单元密度，应考虑以下几种设计因素:

- 维数不同。
- 一个或多个维的详细程度不同。
- 表空间的块（扩展数据块）大小和页大小不同。

执行下列步骤来获得最佳设计:

1. 标识候选维。

   确定哪些查询将采用块级别集群。检查具有下列某些特征或所有特征的列可能具有的工作负载:

   - 任何 IN 列表谓词的范围和等效性
   - 数据的转入或转出
   - Group-by 和 order-by 子句
   - Join 子句（特别是在星型模式环境中）。

2. 估计单元数目。

   标识在根据一组候选维组织的表中可能存在多少个潜在的单元。确定数据中出现的维值的唯一组合的数目。如果表存在，则只需要选择每个列中将为表的维的单值的数目来确定当前数据的准确数目。或者，如果只有表的统计信息，则可以通过乘以候选维的列基数来确定一个大概数目。

   注: 如果表处于分区数据库环境中，并且分布键与考虑的任何维都无关，则必须通过用所有数据除以数据库分区数来确定每个单元的平均数据量。

3. 估计空间占用率或密度。

   按平均情况来说，认为每个单元都有一个部分填充的块，该块中只存储了很少的行。当每个单元的行数变得越来越少时，部分填充的块就会更多。另外还要注意，按平均情况来说（假定存在很少甚至没有数据偏差），可以通过用表中的记录数除以单元数来计算每个单元的记录数。但是，如果表处于分区数据库环境中，则需要考虑每个数据库分区上的每个单元中有多少条记录，因为是按照数据库分区来为数据分配块的。当估计分区数据库环境中的空间占用率和密度时，需要考虑每个数据库分区（而不是整个表）上每个单元的平均记录数。有关更多信息，请参阅『多维集群（MDC）表的创建、布置和使用』一节。

   可以采用以下几种方法来提高密度:

   - 减小块大小，以便使部分填充的块占用较少的空间。

     通过适当减小扩展数据块大小来减小每个块的大小。具有部分填充的块的每个单元或者只包含具有很少记录的一个块的每个单元都占用较少的空间。但是，采用一种折衷办法就是，对于具有很多记录的那些单元，需要更多的块来包含它们。这将增加块索引中的这些单元的块标识（BID）的数目，使这些索引更大并且潜在地导致更多的索引插入和删除操作，因为会更快地清空和填充块。与数目较少的集群数据的较大组合相比较，它还会导致表中的集群数据的更多更小的组合，以装入更多填充单元值。

   - 通过减少维数或者通过增大具有生成列的单元的详细程度来减少单元数。

     可以将一个或多个维上滚为较低的详细程度，以给予它较低的基数。例如，可以仍然根据 Region 和 Product 对将前一个示例中的数据库进行集群，但将 Day 维替换为 YearAndMonth 维。这将为 YearAndMonth、Region 和 Product 维给予基数 60（12 个月乘以 5 年）、12 和 5，可能的单元数为 3600。于是，每个单元将具有更大范围的值，并且降低了只包含很少记录的可能性。

还应该考虑对涉及到的列通常使用的谓词，例如，使用得较多的是 Month of Date、Quarter 或 Day。这将影响更改维的详细程度的期望度。例如，如果大多数谓词是针对特定日期的，并且您根据“月份”对表进行了集群，则 DB2 数据库 Linux 版、UNIX 版和 Windows 版可以使用 YearAndMonth 的块索引来快速缩小包含想要的日期的月份的范围并且仅访问这些相关联的块。但是，在扫描块时，必须应用 Day 谓词来确定哪些日期合格。但是，如果根据 Day 进行集群（并且 Day 具有很高的基数），则可以使用 Day 的块索引来确定扫描哪些块，并且只需要对合格的每个单元的第一条记录重新应用 Day 谓词。在此情况下，在使用用户定义的函数上滚 Region 列（它包含 12 个不同的值）至 Regions West、North、South 和 East 时最好考虑上滚其他维之一以增大单元的密度。

相关概念:

- 『设计顾问程序』（*《性能指南》*）
- 第 183 页的『多维集群（MDC）表创建、布置和使用』
- 第 159 页的『多维集群表』

多维集群（MDC）表创建、布置和使用

当创建 MDC 表时应该考虑许多因素。下列各节讨论了您当前所处的数据库环境（例如，是否具有分区数据库）和为 MDC 表选择的维对于您创建、布置和使用 MDC 表将产生的影响。还讨论了“DB2 设计顾问程序”以及可以如何使用它来提供有关这样一些问题的建议。

将数据从现有表移至多维集群（MDC）表:

要提高数据仓库或大型数据库环境中的查询性能和减少数据维护操作的开销，可以将数据从常规表移至多维集群（MDC）表。要将数据从现有表移至 MDC 表：导出数据，删除原始表（可选），创建多维集群（MDC）表（使用带有 ORGANIZE BY DIMENSIONS 子句的 CREATE TABLE 语句），并将您自己的数据装入 MDC 表。

可以使用称为 SYSPROC.ALTOBJ 的 ALTER TABLE 过程来将现有表中的数据转换为 MDC 表中的数据。可以从“DB2 设计顾问程序”中调用此过程。在这两个表之间转换数据所需要的时间可能很长，这取决于表的大小以及需要转换的数据量。

当改变表时，ALTOBJ 过程将执行下列操作:

- 删除表的所有从属对象
- 重命名表
- 使用新定义来创建表
- 重新创建表的所有从属对象
- 将表中的现有数据变换为新表中需要的数据。也就是说，从旧表中选择数据，然后将该数据装入新表中，可以使用列函数来将旧的数据类型变换为新的数据类型。

SMS 表空间中的多维集群（MDC）表:

如果打算将 MDC 表存储在 SMS 表空间中，则强烈建议使用多页文件分配。

注: 对于在版本 8.2 和更新版本中新创建的数据库，多页文件分配是缺省值。

之所以提出这种建议是因为 MDC 表总是按整个扩展数据块来扩展的，这些扩展数据块中的所有页在物理上保持连续是很重要的。因此，禁用多页文件分配不利于利用空间；而且，启用多页文件分配将显著提高每个扩展数据块中的页在物理上是连续的可能性。

DB2 设计顾问程序上的 MDC 顾问程序功能部件:

“DB2 设计顾问程序”（db2advis）（以前称为“索引顾问程序”）具有 MDC 功能部件。此功能部件建议用于 MDC 表中的集群维（包括基本列的粗糙度）以便提高工作负载性能。*粗糙度*这个术语表示用来减小集群维的基数（单值的数目）的一个算术表达式。粗糙度的常见示例是日期，日期的粗糙度可为日期、日期所在的星期、日期所在的月份或一年中的季度。

使用“DB2 设计顾问程序”的 MDC 功能部件要求数据库中至少存在几个扩展数据块的数据。“DB2 设计顾问程序”使用数据来对数据密度和基数建立模型。

如果数据库的表中没有数据，则“DB2 设计顾问程序”不会建议使用 MDC，即使该数据库包含空表，但有一组虚假的统计信息来表示它是一个已填充的数据库。

建议还标识了用来定义维的粗糙度的潜在生成列。建议中不包括可能的块大小。在为 MDC 表提供建议时，使用表空间的扩展数据块大小。假定将在现有表所在的表空间中创建建议的 MDC 表，因此该表将具有相同的扩展数据块大小。对 MDC 维的建议将根据表空间的扩展数据块大小变化而变化，这是因为扩展数据块大小将影响可以填充到块或单元中的记录数。这将直接影响单元的密度。

只考虑单列维而不考虑组合列维，尽管可以为表建议单个维或多个维。MDC 功能部件将建议大多数受支持的数据类型使用的粗糙度，目标是减小所采用的 MDC 解决方案中的单元的基数。异常的数据类型包括：CHAR、VARCHAR、GRAPHIC 和 VARGRAPH 数据类型。所有受支持的数据类型都将被强制类型转换为 INTEGER 并通过生成的表达式来设置粗糙度。

“DB2 设计顾问程序”的 MDC 功能部件的目标是选择可提高性能的 MDC 解决方案。另一个目标是将数据库的存储器扩充限制在适当的级别。使用统计方法来确定每个表的最大存储器扩充。

顾问程序中的分析操作既会利用块索引访问的优点，也会受到 MDC 对表的维执行插入、更新和删除操作的影响。对表执行这些操作时可能会导致在各个单元之间移动记录。分析操作还会模拟对特定 MDC 维上的数据进行组织时产生的任何表扩充而对性能造成的潜在影响。

通过对 db2advis 实用程序使用 `-m <advise type>` 标志来启用 MDC 功能部件。使用“C”建议类型来指示多维集群表。建议类型为：“I”表示索引、“M”表示具体化查询表、“C”表示 MDC，而“P”表示分区数据库环境。建议类型可以相互组合使用。

注: “DB2 设计顾问程序”将不会处理小于 12 个扩展数据块大小的表。

当提出建议时，顾问程序将同时分析 MQT 和常规基本表。

MDC 功能部件的输出包括:

- 每个表的生成列表达式（用于对 MDC 解决方案中出现的维设置粗糙度）。
- 为每个表建议的 ORGANIZE BY 子句。

对标准输出和作为说明工具的一部分的 ADVISE 表报告了建议。

多维集群（MDC）表和分区数据库环境:

多维集群可以与分区数据库环境联合使用。实际上，MDC 可以作为分区数据库环境的补充。分区数据库环境用来将一个表中的数据分配到多个物理节点或逻辑节点上，以便达到下列目的:

- 利用多台机器来并行增加处理请求。
- 增大表的物理大小（超过单个数据库分区的限制）。
- 提高数据库的可伸缩性。

分发表的原因与该表是 MDC 表还是常规表无关。例如，选择用来组成分布键的列的规则是相同的。MDC 表的分布键可以包括任何列，无论这些列是否组成表的维的一部分。

如果分布键与表的某个维完全相同，则每个数据库分区都将包含该表的不同部分。例如，如果作为示例的 MDC 表按颜色分发在两个数据库分区上，则将使用 Color 列来划分数据。因此，可能在一个数据库分区上找到 Red 和 Blue 片，而在另一个数据库分区上找到 Yellow 片。如果分布键与表中的维不完全相同，则每个数据库分区都将具有每个片的数据的子集。当选择维和估计单元占用率时（请参阅『单元密度』一节），注意，平均来说，每个单元的数据总量是通过用所有数据除以数据库分区数来确定的。

具有多个维的多维集群（MDC）表:

如果知道将在查询中大量使用特定谓词，则可以使用 ORGANIZE BY DIMENSIONS 子句根据涉及到的列对表进行集群。

示例 1:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, c3, c4)
```

示例 1 中的表根据形成逻辑立方体（即，具有三个维）的三个本地列中的值集群。现在可以在查询处理期间根据一个或多个维对表进行逻辑分片，以便涉及的关系运算符仅处理相应的片或单元中的块。注意，块的大小（页数）将是表的扩展数据块大小。

具有基于多列的维的多维集群（MDC）表:

每个维可以由一列或多列组成。作为一个示例，可以创建一个根据包含两列的一个维来集群的表。

示例 2:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, (c3, c4))
```

在示例 2 中，表将根据两个维 c1 和 (c3, c4) 来进行集群。这样，在查询处理期间，表可以根据 c1 维或组合 (c3, c4) 维逻辑分片。该表将与示例 1 中的表具有相同数目的块，但是少一个维块索引。在示例 1 中，将有三个维块索引，列 c1、c3 和 c4 各一个。在示例 2 中，将有两个维块索引，一个针对列 c1，而另一个针对 c3 和 c4。这两个方法的主要不同之处在于，在示例 1 中，仅涉及 c4 的查询可以使用 c4 的维块索引来快速直接地访问相关数据块。在示例 2 中，c4 是维块索引中的辅助键部分，因此仅涉及 c4 的查询涉及更多的处理。但是，在示例 2 中，DB2 数据库 Linux 版、UNIX 版和 Windows 版将少维护和存储一个块索引。

“DB2 设计顾问程序”将不对包含多列的维提供建议。

将列表达式作为维的多维集群（MDC）表:

列表达式也可用于集群维。根据列表达式集群的功能对于将维上滚至更低的详细程度非常有用，例如，将地址上滚为地理位置或区域，或者将日期上滚为星期、月份或年。要以此方式实现维的上滚，可以使用生成列。此类型的列定义将允许使用可表示维的表达式创建列。在示例 3 中，该语句创建根据一个基本列和两个列表达式进行集群的表。

示例 3:

```
CREATE TABLE T1(c1 DATE, c2 INT, c3 INT, c4 DOUBLE,
   c5 DOUBLE GENERATED ALWAYS AS (c3 + c4),
   c6 INT GENERATED ALWAYS AS (MONTH(C1))
      ORGANIZE BY DIMENSIONS (c2, c5, c6)
```

在示例 3 中，c5 列是基于 c3 和 c4 列的表达式，而 c6 列会将 c1 列上滚至更低的详细程度。此语句将根据 c2、c5 和 c6 列中的值集群该表。

对生成列维的范围查询需要单调列函数:

表达式必须是单调的才能为生成列的维派生范围谓词。如果对生成列创建维，则对基本列的查询将能够利用生成列的块索引来提高性能（有一种情况例外）。要使基本列（例如，日期）的范围查询对维块索引使用范围扫描，用来在 CREATE TABLE 语句中生成列的表达式必须是单调的。尽管列表达式可以包括任何有效表达式（包括用户定义的函数（UDF）），但是如果表达式不是单调的，则当等价谓词或 IN 谓词都在基本列上时，它们才能够使用块索引来满足查询。

作为一个示例，假定使用生成列 month 的维来创建 MDC 表，其中 `month = INTEGER (date)/100`。对于该维（month）的查询，可以执行块索引扫描。对于基本列（date）的查询，也可以执行块索引扫描来缩小要扫描的块的范围，然后只将日期的谓词应用于这些块中的行。

编译器将生成要在块索引扫描中使用的其他谓词。例如，对于以下查询:

```
SELECT * FROM MDCTABLE WHERE DATE > "19999/03/03" AND DATE < "2000/01/15"
```

编译器将生成以下谓词: “month >= 199903”和“month < 200001”，它们可以用作维块索引扫描的谓词。当对获得的块进行扫描时，会将原始谓词应用于这些块中的行。

非单调表达式将只允许对该维应用等价谓词。非单调函数的一个较好的示例是 MONTH( )，如示例 3 中的 c6 列的定义所示。如果 c1 列是日期、时间戳记或日期或时间戳记的有效字符串表示法，则函数将返回范围是 1 到 12 的整数值。尽管函数的输出是确定的，但是实际上它生成的输出与阶跃函数（即，循环模式）相似:

```
MONTH(date('99/01/05')) = 1
MONTH(date('99/02/08')) = 2
MONTH(date('99/03/24')) = 3
MONTH(date('99/04/30')) = 4
...
MONTH(date('99/12/09')) = 12
MONTH(date('00/01/18')) = 1
MONTH(date('00/02/24')) = 2
...
```

尽管此示例中的日期是连续增加的，但是 MONTH(date) 不会增加。更具体地说，每当 date1 大于 date2，并不能保证 MONTH(date1) 大于或等于 MONTH(date2)。这是单调性所要求的。此非单调性是允许的，但是它限制了维，基本列的范围谓词不能生成维的范围谓词。但是，表达式的范围谓词是可以的，例如，`where month(c1) between 4 and 6`。这可以采用常规方式使用维的索引，起始键为 4 而停止键为 6。

要使此函数单调，必须将年包括为月份的高位部分。DB2 V9.1 提供对 INTEGER 内置函数的扩展以帮助根据日期定义单调表达式。INTEGER(date) 返回日期的整数表示法，可以分开查找年和月份的整数表示法。例如，`INTEGER(date('2000/05/24'))` 返回 `20000524`，因此 `INTEGER(date('2000/05/24'))/100 = 200005`。函数 INTEGER(date)/100 是单调的。

相似的，内置函数 DECIMAL 和 BIGINT 也具有扩展，所以可以派生单调函数。DECIMAL(timestamp) 返回时间戳记的十进制表示法，可以在单调表达式中使用它来派生月份、天、小时或分钟等等的增加的值。BIGINT(date) 返回日期的大整数表示法，类似于 INTEGER(date)。

只要可能，DB2 会在为表创建生成列或者根据维子句中的表达式创建维时确定表达式的单调性。特定函数被识别为保留单调性，例如，DATENUM( )、DAYS() 和 YEAR( )。并且，列和常量的各种算术表达式，例如，除法、乘法或加法是保留单调性的。当 DB2 确定表达式不保留单调性时，或者如果它不能确定这一点，该维将仅支持对其基本列使用等价谓词。

相关概念:

- 『装入数据时的多维集群注意事项』（*《数据移动指南和参考》*）
- 第 176 页的『设计多维集群（MDC）表』
- 第 134 页的『扩展数据块大小』
- 第 159 页的『多维集群表』

相关任务:

- 『对表定义维』（*《管理指南：实施》*）

相关参考:

- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）
- 『db2empfa - Enable multipage file allocation command』（*Command Reference*）

第 6 章 设计分区数据库

本章讨论使用分区数据库时与管理事务相关的问题。首先讨论更新单个数据库。之后讨论更复杂的情况，即关于使用多个数据库的事务的情况。同时介绍事务管理器的概念，以及从主机或 iSeries 更新数据库时的注意事项。本章还讨论了管理多站点更新时的两阶段落实和使用两阶段落实处理事务时的错误恢复。

在事务中更新单个数据库

事务的最简单形式是在单个工作单元中只对一个数据库读取和写入。此类数据库访问称为*远程工作单元*。

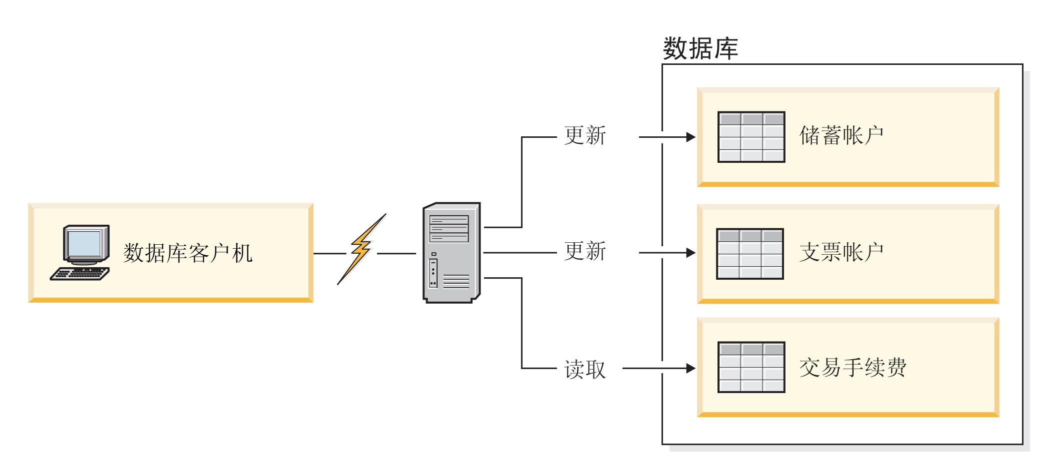


图 60. 在一个事务中使用单个数据库

图 60 显示了一个运行转帐应用程序的数据库客户机，该应用程序访问包含支出帐户和储蓄帐户表以及银行手续费表的数据库。该应用程序必须:

- 接受从用户界面指定的转帐金额
- 从储蓄帐户中扣除该金额，然后确定新的余额
- 读取手续费表，以确定含有给定余额的储蓄帐户的交易手续费
- 从储蓄帐户中扣除交易手续费
- 将转帐的金额添加至支出帐户
- 落实该事务（工作单元）

过程:

要设置这样的应用程序，准备在环境中执行事务时必须进行下列操作:

1. 在同一个数据库中为储蓄帐户、支出帐户和银行业务手续费表创建表
2. 若物理上是远程的，则设置数据库服务器以使用相应的通信协议
3. 若物理上是远程的，则编目节点和数据库以便在数据库服务器上标识该数据库
4. 预编译应用程序以指定类型 1 连接；即在 PRECOMPILE PROGRAM 命令上指定 CONNECT 1（缺省值）

相关概念:

- 第 21 页的『工作单元』

相关任务:

- 第 190 页的『在多数据库事务中更新单个数据库』
- 第 191 页的『在事务中更新多个数据库』

相关参考:

- 『PRECOMPILE command』（*Command Reference*）

在单个事务中使用多个数据库

当在单个事务中使用多个数据库时，对设置和管理环境的要求是不同的，这取决于在该事务中要更新的数据库的数量。以下主题讨论这些要求。

在多数据库事务中更新单个数据库

若您的数据分布在多个数据库上，您可能希望在从一个或多个数据库读取的同时更新一个数据库。可以在单个工作单元（事务）中执行此类访问。

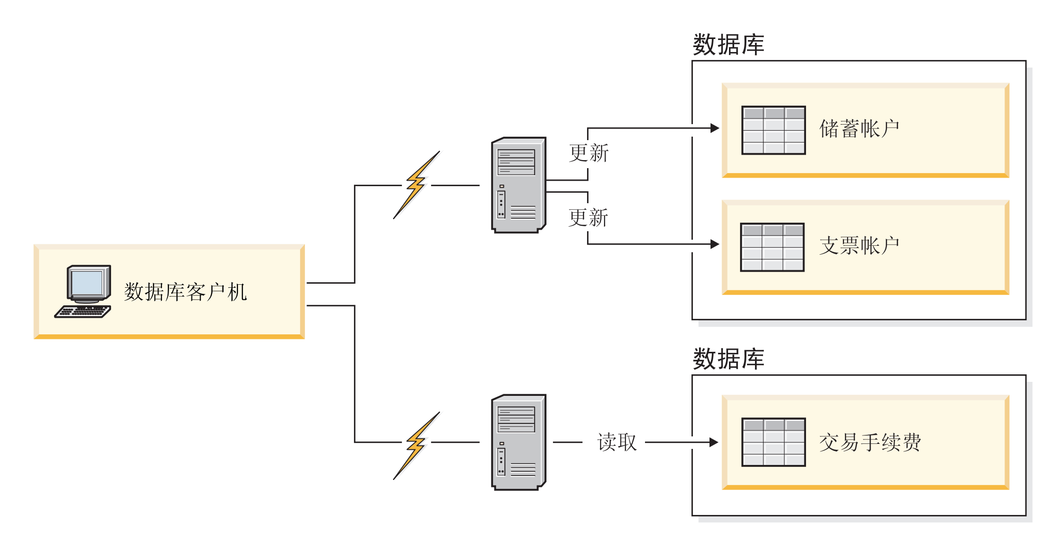


图 61. 在单个事务中使用多个数据库

图 61 显示了一个运行转帐应用程序的数据库客户机，该应用程序访问两个数据库服务器：一个包含支票和储蓄帐户，另一个包含银行业务或交易手续费支付表。

过程:

要为此环境设置转帐应用程序，必须:

1. 在相应数据库中创建必需的表
2. 若物理上是远程的，则设置数据库服务器以使用相应的通信协议
3. 若物理上是远程的，则编目节点和数据库，以便在数据库服务器上标识这些数据库
4. 预编译应用程序以指定类型 2 连接（即在 PRECOMPILE PROGRAM 命令上指定 CONNECT 2）和一阶段落实（即在 PRECOMPILE PROGRAM 命令上指定 SYNCPOINT ONEPHASE）

如果数据库位于主机或 iSeries 数据库服务器上，则需要 DB2 Connect™ 才能与这些服务器连接。

相关概念:

- 第 21 页的『工作单元』

相关任务:

- 第 189 页的『在事务中更新单个数据库』
- 第 191 页的『在事务中更新多个数据库』

相关参考:

- 『PRECOMPILE command』（*Command Reference*）

在事务中更新多个数据库

若您的数据分布在多个数据库上，则您可能想在单个事务中读取和更新多个数据库。此类数据库访问称为*多站点更新*。

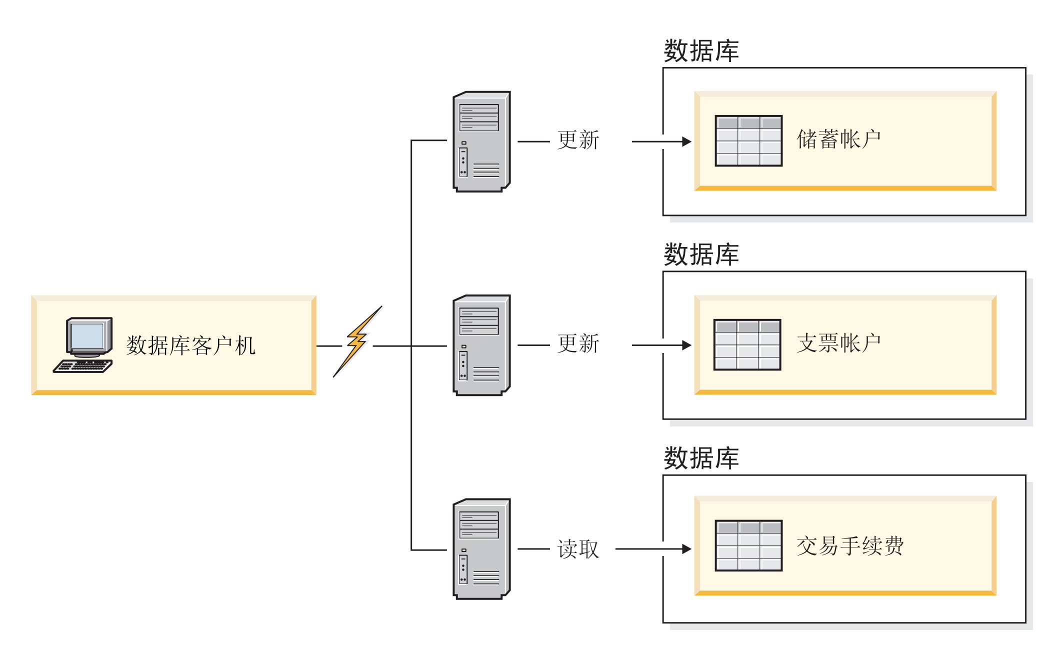


图 62. 在单个事务中更新多个数据库

图 62 显示了一个运行转帐应用程序的数据库客户机，该应用程序访问三个数据库服务器：一个包含支出帐户，另一个包含储蓄帐户，第三个包含银行业务手续费表。

过程:

要为此环境设置转帐应用程序，您有两种选择:

1. 使用 DB2 事务管理器（TM）:
   a. 在相应数据库中创建必需的表
   b. 若物理上是远程的，则设置数据库服务器以使用相应的通信协议
   c. 若物理上是远程的，则编目节点和数据库，以便在数据库服务器上标识这些数据库

d. 预编译应用程序以指定类型 2 连接（即在 PRECOMPILE PROGRAM 命令上指定 CONNECT 2）和两阶段落实（即在 PRECOMPILE PROGRAM 命令上指定 SYNCPOINT TWOPHASE）
e. 配置 DB2 事务管理器（TM）。

2. 使用符合 XA 的事务管理器:
   a. 在相应数据库中创建必需的表
   b. 若物理上是远程的，则设置数据库服务器以使用相应的通信协议
   c. 若物理上是远程的，则编目节点和数据库，以便在数据库服务器上标识这些数据库
   d. 预编译应用程序以指定类型 2 连接（即在 PRECOMPILE PROGRAM 命令上指定 CONNECT 2）和一阶段落实（即在 PRECOMPILE PROGRAM 命令上指定 SYNCPOINT ONEPHASE）
   e. 配置符合 XA 的事务管理器以使用 DB2 数据库。

相关概念:

- 第 192 页的『DB2 事务管理器』
- 第 21 页的『工作单元』

相关任务:

- 第 190 页的『在多数据库事务中更新单个数据库』
- 第 189 页的『在事务中更新单个数据库』

相关参考:

- 『PRECOMPILE command』（*Command Reference*）

DB2 事务管理器

DB2 数据库 Linux 版、UNIX 版和 Windows 版事务管理器（TM）向事务指定标识、监视它们的进程，并负责处理事务的完成和失败。DB2 数据库系统和 DB2 Connect 提供事务管理器。DB2 TM 在指定的 TM 数据库中存储事务信息。

数据库管理器提供事务管理器功能，这些功能可用于协调在单个工作单元内几个数据库的更新。数据库客户机自动协调工作单元，并使用*事务管理器数据库*来注册每个事务并跟踪其完成状态。

可以将 DB2 事务管理器与 DB2 数据库配合使用。如果有不同于 DB2 数据库的资源想要参与两阶段落实事务，则可以使用符合 XA 的事务管理器。

相关概念:

- 第 192 页的『DB2 数据库事务管理器配置』
- 第 195 页的『两阶段落实』
- 第 21 页的『工作单元』

DB2 数据库事务管理器配置

如果正在使用符合 XA 的事务管理器，例如，IBM® WebSphere®、BEA Tuxedo 或 Microsoft Transaction Server，则应遵循该产品的配置指示信息。

当使用 DB2 数据库 Linux 版、UNIX 版和 Windows 版来协调事务时，必须满足特定配置需求。若您只使用 TCP/IP 来进行通信，且您的事务中仅涉及 DB2 数据库 Linux 版、UNIX 版和 Windows 版或 DB2 通用数据库 iSeries V5 版、z/OS 版或 OS/390 版这些数据库服务器，则配置会简单得多。

DB2 Connect 不再支持对主机或 iSeries 服务器进行 SNA 两阶段落实访问。

使用 TCP/IP 连接的 DB2 数据库 Linux 版、UNIX 版和 Windows 版和 DB2 通用数据库 z/OS 版、OS/390 版和 iSeries V5 版

若以下所有陈述都适合于您的环境，则多站点更新的配置步骤就会简化。

- 与远程数据库服务器（包括 DB2 UDB z/OS 版、OS/390 版和 iSeries V5 版）的所有通信都只使用 TCP/IP。
- DB2 数据库 Linux 版、UNIX 版和 Windows 版或 DB2 通用数据库 z/OS 版、OS/390 版或 iSeries V5 版是事务中涉及的唯一数据库服务器。

将用作事务管理器数据库的数据库由数据库客户机的数据库管理器配置参数 *tm_database* 确定。当设置此配置参数时，考虑下列因素:

- 事务管理器数据库可以是:
  - DB2 通用数据库 UNIX 版或 Windows 版的版本 8 数据库
  - DB2 z/OS 和 OS/390 版的版本 7 数据库或 DB2 OS/390 版的版本 5 或版本 6 数据库
  - DB2 iSeries V5 版数据库

  建议使用 DB2 z/OS 版、OS/390 版和 iSeries V5 版数据库服务器来作为事务管理器数据库。z/OS、OS/390 和 iSeries V5 系统通常比工作站服务器更安全，降低了意外断电和重新引导等的可能性。因此，在再同步时使用的恢复日志更安全。
- 若对 *tm_database* 配置参数指定了值 `1ST_CONN`，则与应用程序连接的第一个数据库被用作事务管理器数据库。

  当使用 `1ST_CONN` 时，必须小心。仅当能够比较容易地确保所有参与的数据库都正确编目时，才应使用此配置，即，如果启动该事务的数据库客户机位于包含参与数据库（包括事务管理器数据库）的同一个实例中。
- 如果使用 TCP/IP V6，则将根据所选择的操作系统配置方式创建 IP 地址。
- 如果使用的是“自动配置”方式，则将从 IPv6 地址中抽取 MAC 地址，并将它用在内部 DB2 协调程序的工作单元标识中。不需要更改任何配置。
- 如果使用的是“手工配置”方式，则使用 IPv6 地址的最后 6 个字节创建内部 DB2 协调程序的工作单元标识。为了避免出现冲突，用户必须确保 IPv6 地址的最后 6 个字节在网络中是唯一的。

注:

1. DB2 协调程序是 DB2 客户机，必须在具有 DB2 客户机的系统上执行配置更改。
2. 若您的应用程序试图与正用作事务管理器数据库的数据库断开连接，则将接收到警告消息，且该连接将被挂起，直到落实该工作单元为止。

事务管理器的配置参数

当您设置环境以支持事务管理器时，应该考虑下列配置参数。

数据库管理器配置参数

- *tm_database*

  此参数标识每个 DB2 实例的“事务管理器”（TM）数据库的名称。

- *spm_name*

  此参数向数据库管理器标识 DB2 Connect 同步点管理器实例的名称。为了使再同步成功，该名称在您的网络中必须是唯一的。

- *resync_interval*

  此参数标识一个时间间隔（以秒计），在该时间间隔之后，“DB2 事务管理器”、DB2 服务器数据库管理器以及 DB2 Connect 同步点管理器或 DB2 同步点管理器应该重新尝试恢复任何未完成的不确定事务。

- *spm_log_file_sz*

  此参数指定 SPM 日志文件的大小（以 4 KB 页计）。

- *spm_max_resync*

  此参数标识可同时执行再同步操作的代理程序数。

- *spm_log_path*

  此参数标识 SPM 日志文件的日志路径。

数据库配置参数

- *maxappls*

  此参数指定允许的最大活动应用程序数。它的值必须等于或大于连接的应用程序数、可能在完成一个两阶段落实或回滚的过程中并存的应用程序的数目以及预期任何时候可能存在的不确定事务的数目之和。

- *autorestart*

  此数据库配置参数指定在需要时是否自动调用 RESTART DATABASE 例程。缺省值是 YES（即，启用）。

  包含不确定事务的数据库需要重新启动数据库操作才能启动。若断开与该数据库的上一个连接时未启用 *autorestart*，则下一个连接将失败并需要显式的 RESTART DATABASE 调用。这种情况将一直存在，直到事务管理器的再同步操作或管理员启动的试探性操作除去了不确定事务为止。当发出 RESTART DATABASE 命令时，若该数据库中有任何不确定事务，将返回一条消息。管理员就可以使用 LIST INDOUBT TRANSACTIONS 命令和其他命令行处理器（CLP）命令来获取有关那些不确定事务的信息。

相关概念:

- 第 192 页的『DB2 事务管理器』

相关任务:

- 第 221 页的『配置 BEA Tuxedo』

- 第 220 页的『配置 IBM TXSeries CICS』
- 第 220 页的『配置 IBM TXSeries Encina』
- 第 219 页的『配置 IBM WebSphere Application Server』

相关参考:

- 『autorestart – 启用自动重新启动配置参数』(*《性能指南》*)
- 『maxappls – 最大活动应用程序数配置参数』(*《性能指南》*)
- 『spm_log_path – 同步点管理器日志文件路径配置参数』(*《性能指南》*)
- 『spm_log_file_sz – 同步点管理器日志文件大小配置参数』(*《性能指南》*)
- 『spm_name – 同步点管理器名称配置参数』(*《性能指南》*)
- 『spm_max_resync – 同步点管理器再同步代理程序限制配置参数』(*《性能指南》*)
- 『tm_database – 事务管理器数据库名称配置参数』(*《性能指南》*)
- 『resync_interval – 事务再同步时间间隔配置参数』(*《性能指南》*)

从主机或 iSeries 客户机更新数据库

在主机或 iSeries 客户机上执行的应用程序可以访问位于 DB2 数据库 Linux 版、UNIX 版和 Windows 版数据库服务器上的数据。TCP/IP 是用于此访问的唯一协议。所有平台上的 DB2 数据库 Linux 版、UNIX 版和 Windows 版服务器不再支持从远程客户机进行 SNA 访问。

在版本 8 之前，从主机或 iSeries 客户机进行 TCP/IP 访问只支持一阶段落实访问。DB2 数据库 Linux 版、UNIX 版和 Windows 版现在允许从主机或 iSeries 客户机进行 TCP/IP 两阶段落实访问。当使用 TCP/IP 两阶段落实访问时，不需要使用“同步点管理器”(SPM)。

DB2 TCP/IP 侦听器在服务器上必须是活动的，主机或 iSeries 客户机才能访问它。可以通过使用 db2set 命令来验证注册表变量 DB2COMM 的值为“tcpip”来检查 TCP/IP 侦听器是否是活动的；并使用 GET DBM CFG 命令来检查数据库管理器配置参数 **svcename** 被设置为服务名称。如果侦听器不是活动的，则可以使用 db2set 命令和 UPDATE DBM CFG 命令来使它变得活动。

相关参考:

- 『spm_name – 同步点管理器名称配置参数』(*《性能指南》*)
- 『通信变量』(*《性能指南》*)

两阶段落实

第 196 页的图 63 举例说明了多站点更新包括的步骤。了解一个事务是如何管理的将有助于您在两阶段落实过程期间发生错误时解决问题。

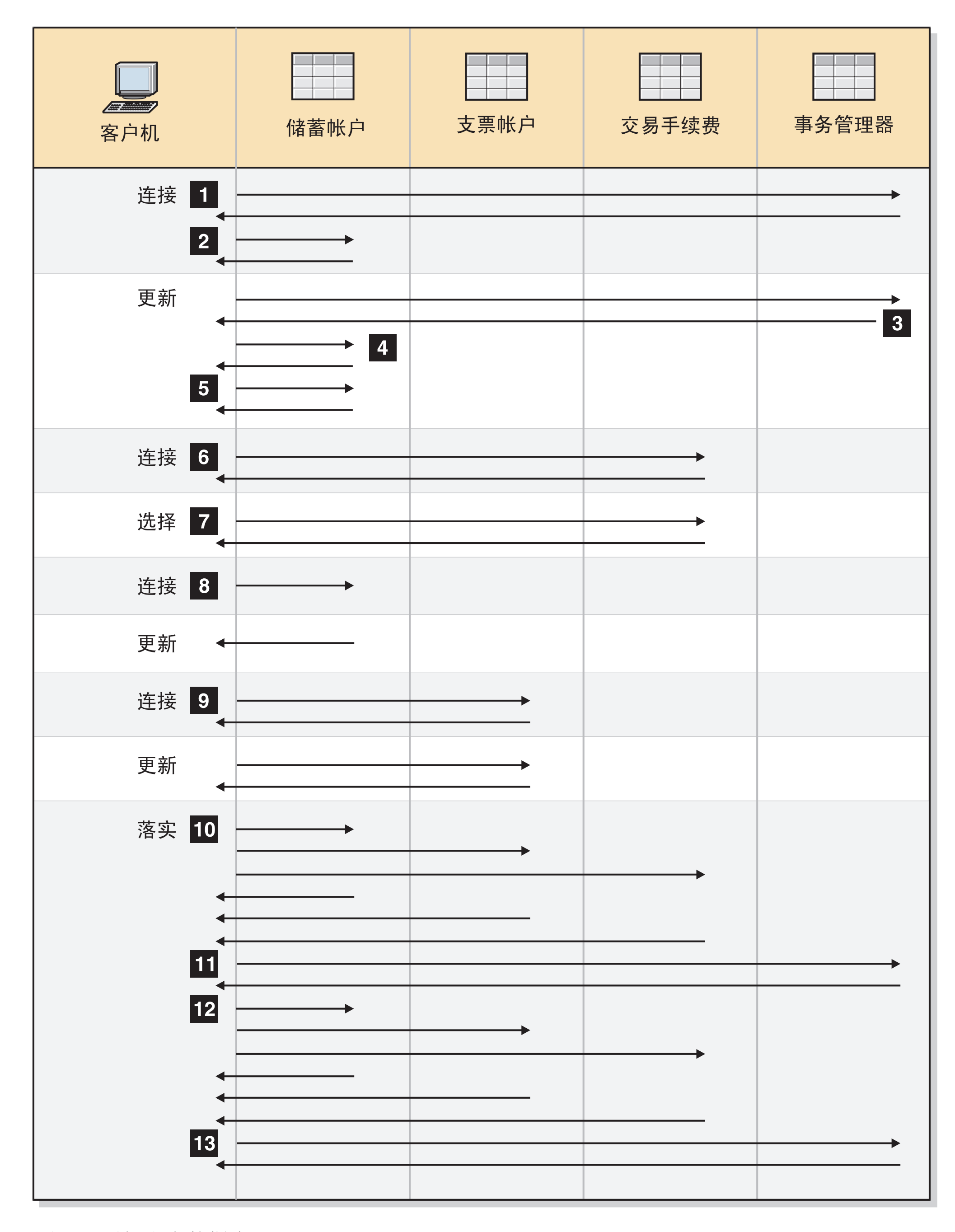


图 63. 更新多个数据库

0 为两阶段落实准备该应用程序。此过程可以通过预编译选项来完成。此过程还可以通过 DB2 数据库 Linux 版、UNIX 版和 Windows 版 CLI（调用级接口）配置来完成。

1 当数据库客户机要与 SAVINGS_DB 数据库连接时，它首先在内部与事务管理器（TM）数据库连接。该 TM 数据库将确认返回至数据库客户机。若将数据库管理器配置参数 *tm_database* 设置为 1ST_CONN，则 SAVINGS_DB 在此应用程序实例的持续时间内成为事务管理器数据库。

2 建立与 SAVINGS_DB 数据库的连接，并确认该连接。

3 数据库客户机开始更新 SAVINGS_ACCOUNT 表。这开始工作单元。TM 数据

库提供该工作单元的一个事务标识，以响应数据库客户机。注意，当运行该工作单元中的第一条 SQL 语句时，而不是在建立连接期间，要对该工作单元进行注册。

4 当接收到该事务标识之后，数据库客户机向包含 SAVINGS_ACCOUNT 表的数据库注册该工作单元。一个响应将发送回客户机，以指示该工作单元已成功注册。

5 以正常方式处理对 SAVINGS_DB 数据库发出的 SQL 语句。当使用在一个程序中嵌入的 SQL 语句时，对每条语句的响应都会在 SQLCA 中返回。

6 当在工作单元内第一次访问该数据库期间，在包含 TRANSACTION_FEE 表的该 FEE_DB 数据库中注册该事务标识。

7 以正常方式处理对 FEE_DB 数据库发出的任何 SQL 语句。

8 可以适当设置该连接，来对 SAVINGS_DB 数据库运行附加的 SQL 语句。由于已向 SAVINGS_DB 数据库注册了该工作单元 **4**，因此数据库客户机不必再次执行注册步骤。

9 连接 CHECKING_DB 数据库，并遵循 **6** 和 **7** 中描述的规则来使用它。

10 当数据库客户机请求落实工作单元时，会将一条*准备*消息发送至参与该工作单元的所有数据库。每个数据库都将一个“PREPARED”记录写至它的日志文件，并答复数据库客户机。

11 在数据库客户机接收到来自所有数据库的肯定响应之后，它会向事务管理器数据库发送一条消息，以通知数据库现在已准备落实工作单元（PREPARED）。该事务管理器数据库则将一个“PREPARED”记录写入它的日志文件，并发送答复，以通知该客户机可以启动落实过程的第二阶段。

12 在落实过程的第二阶段期间，数据库客户机会向所有参与数据库发送一条消息，告知它们要落实。每个数据库则将一个“COMMITTED”记录写入它的日志文件，并释放为此工作单元挂起的锁定。当该数据库完成对更改的落实时，它会将一个答复发送至客户机。

13 在数据库客户机接收到来自所有参与数据库的肯定响应之后，它会向事务管理器数据库发送一条消息，以通知数据库该工作单元已完成。于是，事务管理器数据库将一个“COMMITTED”记录写至其日志文件，指示该工作单元已完成，并答复该客户机，指示它已完成。

相关概念:

- 第 192 页的『DB2 事务管理器』
- 第 21 页的『工作单元』

两阶段落实期间的错误恢复

从错误状态恢复是一个与应用程序编程、系统管理、数据库管理和系统操作相关的正常任务。将数据库分布在几个远程服务器上会增加由于网络或通信故障而导致错误的可能性。要确保数据完整性，数据库管理器提供两阶段落实进程。下面说明数据库管理器如何在两阶段落实过程期间处理错误:

- **第一阶段的错误**

若一个数据库通知“准备落实工作单元失败”，则数据库客户机将在落实过程的第二阶段期间回滚该工作单元。在这种情况下，不会向事务管理器数据库发送准备消息。

在第二个阶段期间，客户机会向所有在第一阶段成功地准备了落实的参与数据库发送一条回滚消息。于是，每个数据库将一个“ABORT”记录写入它的日志文件，并释放为此工作单元挂起的锁定。

- **第二阶段的错误**

  此阶段中的错误处理取决于第二阶段将落实还是回滚该事务。若第一阶段遇到错误，则第二阶段将只回滚该事务。

  若其中一个参与数据库落实工作单元失败（可能由于通信故障），则事务管理器数据库将对失败的数据库重试落实。然而，将通过 SQLCA 通知应用程序落实成功。DB2 数据库 Linux 版、UNIX 版和 Windows 版将确保落实数据库服务器中尚未落实的事务。数据库管理器配置参数 *resync_interval* 用来指定事务管理器数据库在尝试落实工作单元之间要等待的时间。在数据库服务器上挂起所有锁定，直到落实工作单元为止。

  若事务管理器数据库失败，它将在重新启动数据库时使该工作单元再同步。再同步过程将试图完成所有*不确定事务*；即，已完成第一阶段但未完成第二阶段落实过程的那些事务。与事务管理器数据库相关联的数据库管理器通过下列各项执行再同步:

  1. 与指示在落实过程的第一阶段期间“已准备”（PREPARED）落实的数据库连接。
  2. 试图落实那些数据库中的不确定事务。（若找不到不确定事务，则数据库管理器假定在落实过程的第二阶段期间该数据库已成功落实该事务。）
  3. 当在参与数据库中落实了所有不确定事务之后，再落实事务管理器数据库中的不确定事务。

  若其中一个参与数据库失败并重新启动，则此数据库的数据库管理器将查询事务管理器数据库，以获知此事务的状态，来确定是否应该回滚该事务。若在日志中找不到该事务，则数据库管理器假定该事务已被回滚，且将回滚此数据库中的不确定事务。否则，该数据库等待事务管理器数据库发出落实请求。

  若该事务已由事务处理监视器（符合 XA 的事务管理器）协调，则数据库将始终依靠 TP 监视器来启动该再同步。

若由于某种原因您不能等待事务管理器自动解决不确定事务，则您可以执行一些操作来以手工方式解决它们。此手工过程有时称为“作启发式决策”。

当 autorestart=off 时的错误恢复

如果 *autorestart* 数据库配置参数设置为 OFF，且 TM 或 RM 数据库中存在不确定事务，则需要执行 RESTART DATABASE 命令才能启动再同步进程。当从命令行处理器发出 RESTART DATABASE 命令时，将使用不同的会话。若您从同一个会话中重新启动另一个数据库，则先前调用所建立的连接将被断开，并且必须再次重新启动。在 LIST INDOUBT TRANSACTIONS 命令不再返回更多不确定事务之后，发出 TERMINATE 命令来断开该连接。

相关概念:

- 第 195 页的『两阶段落实』

相关任务:

- 第 212 页的『手工解决不确定事务』

相关参考:

- 『autorestart － 启用自动重新启动配置参数』（*《性能指南》*）
- 『LIST INDOUBT TRANSACTIONS command』（*Command Reference*）
- 『RESTART DATABASE command』（*Command Reference*）
- 『TERMINATE command』（*Command Reference*）

第 7 章 针对符合 XA 的事务管理器进行设计

如果有除 DB2 数据库以外的资源，且希望它参与两阶段落实事务，则可能想将您的数据库与符合 XA 的事务管理器配合使用。如果事务仅访问 DB2 数据库，则应使用 DB2 事务管理器（如第 191 页的『在事务中更新多个数据库』中所述）。

本章中的主题将帮助您将数据库管理器与符合 XA 的事务管理器（例如，IBM WebSphere 或 BEA Tuxedo）配合使用。

如果要查找关于 Microsoft Transaction Server 的信息，请参阅 *Call Level Interface Guide and Reference, Volume 1*。

如果您要使用符合 XA 的事务管理器，或正在实施该事务管理器，则可从我们的技术支持 Web站点获取更多信息:

`http://www.ibm.com/software/data/db2/udb/winos2unix/support`

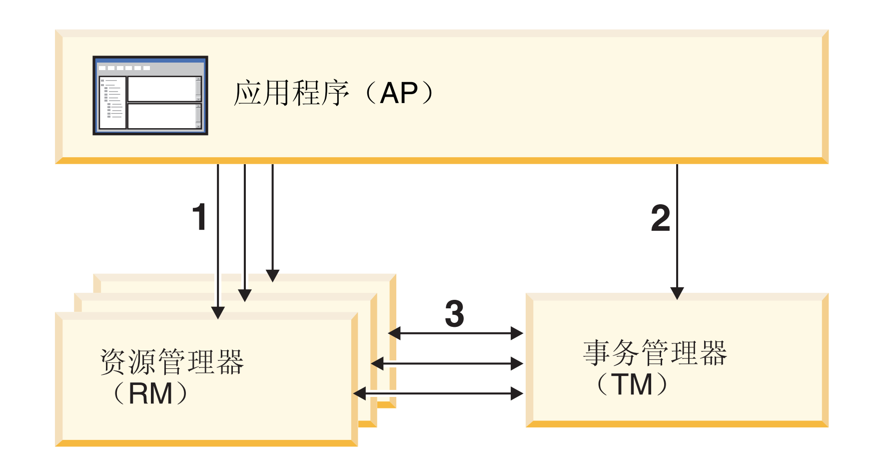
在此页面中选择“DB2”，然后使用关键字“XA”在 Web 站点中搜索关于符合 XA 的事务管理器的最新可用信息。

X/Open 分布式事务处理模型

“X/Open 分布式事务处理”（DTP）模型包括三个相关的组件:

- 应用程序（AP）
- 事务管理器（TM）
- 资源管理器（RM）

第 202 页的图 64 举例说明了此模型，并显示了这些组件之间的关系。



图注

1 — AP 使用一组 RM 中的资源
2 — AP 通过 TM 接口定义事务边界
3 — TM 和 RM 交换事务信息

图 64. X/Open 分布式事务处理（DTP）模型

应用程序（AP）

应用程序（AP）定义事务边界，并定义构成该事务的特定于应用程序的操作。

例如，CICS® 应用程序也许要访问资源管理器（RM），例如，数据库和“CICS 瞬时数据队列”，并使用编程逻辑来处理数据。通过针对该资源管理器的函数调用，将每个访问请求传送至适当的 RM。对于 DB2 产品，这些可能是由 DB2 数据库预编译器为每条 SQL 语句生成的函数调用，或者是程序员直接使用 API 来编码的数据库调用。

事务管理器（TM）产品通常包括用于运行用户应用程序的事务处理（TP）监视器。TP 监视器提供 API 以允许应用程序启动和结束事务，在希望运行该应用程序的多个用户之间执行应用程序的调度和负载均衡。分布式事务处理（DTP）环境中的应用程序实际上是用户应用程序和 TP 监视器的组合。

要实现有效的联机事务处理（OLTP）环境，TP 监视器会在启动时预分配大量服务器进程，然后在多个用户事务之间进行安排并复用它们。这样通过允许使用更少的服务器进程及其对应的 RM 进程来支持更多的并发用户，节省了系统资源。再利用这些进程还避免了为每个用户的事务或程序在 TM 和 RM 中启动一个进程的开销。（一个程序调用一个或多个事务。）这还意味着这些服务器进程对于 TM 和 RM 是真正的“用户进程”。它隐含安全性管理和应用程序编程。

从 TP 监视器可进行下列类型的事务:

- 非 XA 事务

这些事务涉及由于未定义给 TM 而未在 TM 的两阶段落实协议下协调的 RM。若该应用程序需要访问不支持 XA 接口的 RM，这可能是需要的。TP 监视器只是提供有效的应用程序调度和负载均衡。由于 TM 未显式“打开” RM 来进行 XA 处理，所以 RM 将此应用程序作为在非 DTP 环境中运行的任何其他应用程序来处理。

- 全局事务

  这些事务涉及定义给 TM 并在 TM 的两阶段落实控制下的 RM。全局事务是可能涉及一个或多个 RM 的工作单元。*事务分支*是 TM 和 RM 之间支持全局事务的工作的一部分。当通过一个或多个由 TM 协调的应用程序进程来访问多个 RM 时，一个全局事务可能有多个事务分支。

  当多个应用程序进程中的每一个都访问 RM 时，它们就像在单独的全局事务中一样，存在松散耦合的全局事务，但是那些应用程序处于 TM 的协调下。每个应用程序进程在一个 RM 内将有它自己的事务分支。当 AP、TM 或 RM 中任何一个请求落实或回滚时，会一起完成事务分支。该应用程序负责确保在这些分支之间不发生资源死锁。（注意，DB2 事务管理器为使用 SYNCPOINT(TWOPHASE) 选项准备的应用程序执行的事务协调大致等效于这些松散耦合的全局事务。

  当多个应用程序进程轮流在一个 RM 中的相同事务分支下工作时，就会存在紧密耦合的全局事务。对于 RM，这两个应用程序进程是一个单个的实体。RM 必须确保在事务分支内不发生资源死锁。

事务管理器（TM）

事务管理器（TM）向事务指定标识，监视它们的进程，并负责处理事务的完成和失败。事务分支标识（称为 XID）由 TM 指定，以标识一个 RM 内的全局事务和特定分支。它是一个 TM 中的日志与一个 RM 中的日志之间的相关标记。两阶段落实或回滚需要 XID，以便在系统启动时执行*再同步*操作（也称为*再同步*（*resync*）），或在需要时允许管理员执行*试探*操作（也称为*手工干预*）。

在 TP 监视器启动后，它请求 TM 打开一组应用程序服务器已定义的所有 RM。TM 将 **xa_open** 调用传送至 RM，以便可以将它们初始化以进行 DTP 处理。作为此启动过程的一部分，TM 执行再同步以恢复所有的*不确定事务*。不确定事务是处于未确定状态的全局事务。当成功完成两阶段落实协议的第一阶段（即，准备阶段）之后 TM（或至少一个 RM）成为不可用时，就会出现此情况。在 TM 可以使自己的日志与再次变成可用的 RM 的日志相符合之前，RM 不会知道要落实还是回滚它的事务分支。要执行再同步操作，TM 会一次或多次地将 **xa_recover** 调用发出至每个 RM，以标识所有不确定事务。TM 将这些回答与它自己的日志中的信息作比较，以确定应该通知 RM **xa_commit** 那些事务，还是 **xa_rollback** 那些事务。若通过管理员执行试探操作，RM 已经落实或回滚了它的不确定事务的分支，则 TM 向该 RM 发出 **xa_forget** 调用来完成再同步操作。

当用户应用程序请求落实或回滚时，它必须使用 TP 监视器或 TM 提供的 API，以便 TM 可以在所有参与的 RM 之间协调落实和回滚。例如，当 CICS 应用程序发出 CICS SYNCPOINT 请求以落实一个事务时，CICS XA TM（在“Encina 服务器”中实施）将依次发出 XA 调用，如 **xa_end**、**xa_prepare**、 **xa_commit** 或 **xa_rollback**，以请求 RM 落实或回滚该事务。若只涉及一个 RM，或者若 RM 回答它的分支是只读的，则 TM 可选择使用一阶段落实以替代两阶段落实。

资源管理器（RM）

资源管理器（RM）提供对共享资源（如数据库）的访问。

作为数据库的资源管理器的 DB2 系统可以参与由符合 XA 的 TM 协调的*全局事务*。根据 XA 接口的要求，数据库管理器提供 xa_switch_t 类型的 *db2xa_switch* 外部 C 变量，以将 XA 开关结构返回至该 TM。此数据结构包含由该 TM 调用的各种 XA 例程的地址和 RM 的操作特征。

RM 可使用两种方法来对它参与每个全局事务进行注册：*静态注册*和*动态注册*：

- 静态注册要求 TM（为每个事务）将 **xa_start**、**xa_end** 和 **xa_prepare** 等一系列调用发出至为服务器应用程序定义的所有 RM，而不管给定的 RM 是否正由该事务使用。若并非每个事务都包括每个 RM，则这是低效的，并且低效的程度与已定义的 RM 的数目成比例。
- 动态注册（由 DB2 使用）灵活高效。仅当 RM 接收到对其资源的请求时，该 RM 才使用 **ax_reg** 调用向 TM 注册。注意，即使只定义了一个 RM 时，或当每个事务都使用每个 RM 时，此方法也不存在性能方面的缺点，因为 **ax_reg** 和 **xa_start** 调用在 TM 中具有相似的路径。

XA 接口提供 TM 和 RM 之间的双向通信。它是两个 DTP 软件组件之间的系统级接口，而不是应用程序开发者编码的普通应用程序接口。但是，应用程序开发人员应该熟悉该 DTP 软件组件所强加的编程限制。

虽然 XA 接口是不变的，但是每个符合 XA 的 TM 可能具有集成 RM 的产品特定方式。有关将 DB2 产品作为资源管理器与特定的事务管理器集成的信息，请参阅适当的 TM 产品文档。

相关概念:

- 第 215 页的『XA 事务管理器的安全性注意事项』
- 『X/Open XA Interface programming considerations』（*Developing SQL and External Routines*）
- 第 217 页的『DB2 数据库 Linux 版、UNIX 版和 Windows 版支持的 XA 功能』

相关任务:

- 第 191 页的『在事务中更新多个数据库』

资源管理器设置

将每个数据库定义为事务管理器（TM）的一个单独的资源管理器（RM），且必须用 xa_open 字符串标识该数据库。

当将一个数据库设置为资源管理器时，不需要 xa_close 字符串。若提供了此字符串，数据库管理器会忽略它。

数据库连接注意事项

客户机自动重新路由（ACR）

每当服务器崩溃时，连接至该服务器的每台客户机都将产生通信错误，该错误将终止连接，并归结为应用程序错误。在可用性非常重要的应用程序环境中，用户将具有冗

余设置，或者将服务器的故障转移至备用节点。在任何一种情况下，DB2 数据库 Linux 版、UNIX 版和 Windows 版客户机代码都将尝试与原始数据库（它可能正在一个故障转移节点上运行，该节点的 IP 地址也进行故障转移）重新建立连接或者与另一台服务器上的新数据库建立连接。然后，使用 SQLCODE 来通知应用程序以指示已重新路由连接并且已经回滚了正在运行的特定事务。此时，应用程序可以选择是重新运行该事务还是继续下一步。

当使用 ACR 时，已失败的主数据库与“将故障转移至”的备用数据库之间的数据一致性在很大程度上取决于连接被重新路由至的数据库中的数据库日志的状态。为了便于讨论，我们将这种数据库称为“备用数据库”，而将此备用数据库所在的服务器称为“备用服务器”。如果在发生故障时备用数据库是已失败的主数据库的精确副本，则备用数据库中的数据将是一致的，并且将没有数据完整性问题。但是，如果备用数据库不是已失败的主数据库的精确副本，则可能存在数据完整性问题，这是由于“XA 事务管理器”已准备好但是尚未落实的事务的事务结果不一致所造成的。这些事务称为不确定事务。将使用 ACT 功能的“数据库管理员”和应用程序开发者必须意识到使用此功能时产生数据完整性问题的风险。

下面几节描述了各种 DB2 数据库 Linux 版、UNIX 版和 Windows 版环境和每种数据库环境中产生数据完整性问题的风险。

高可用性灾难恢复（HADR）:

DB2 的“高可用性灾难恢复”功能（HADR）可以用来控制在主数据库产生故障之后应用程序重新获得连接时主数据库与备用数据库之间的日志重复级别。用来控制日志重复级别的数据库配置参数称为 *hadr_syncmode*。此参数有三个可能的值:

- SYNC

  在这三种方式中，此方式对防止发生事务丢失提供了最强有力的保护，但是其事务响应时间最长。正如此方式的名称所示，SYNC 用来使主数据库与备用数据库中写入事务日志时同步。当主数据库已写入它自己的日志文件，并且已接收到来自同样已将日志写入备用数据库的备用数据库的确认时，同步就完成了。

  如果使用“XA 事务管理器”来协调涉及到 DB2 资源的事务，则强烈建议使用 SYNC 方式。当客户机被重新路由至备用数据库时，因为它是主数据库的精确副本，所以 SYNC 方式将保证数据完整性以及事务再同步完整性。

- NEARSYNC

  与 SYNC 方式相比，此方式对防止发生事务丢失提供的保护稍微差一点，但是它的事务响应时间较短。仅当主数据库已将日志写入它自己的日志文件，并且已接收到来自同样已将日志写入备用数据库上的主存储器的备用数据库的确认时，主数据库才认为已成功写入日志。如果备用数据库在它可以将日志从内存复制到磁盘之前发生崩溃，则这些日志短期内就会在该备用数据库上丢失。

  假定丢失了数据库日志，并且备用数据库不是主数据库的精确副本，则数据完整性可能会受到影响。如果给定的事务是不确定事务，并且主数据库发生了崩溃，则数据完整性就会受到影响。假定事务结果是 COMMIT。当 XA TM 发出后续 XA_COMMIT 请求时将失败，原因是主数据库已崩溃。因为 XA_COMMIT 请求已失败，所以 XA TM 将需要通过发出 XA_RECOVER 请求来恢复此数据库上的此事务。备用数据库将通过返回它的所有 INDOUBT（不确定）的事务来作出响应。如果在将“内存中”的数据库日志写入磁盘之前，并且在 XA TM 发出 XA_RECOVER

请求之前，备用数据库发生崩溃并且重新启动，则该备用数据库将丢失有关该事务的日志信息，并且不能返回它来作为对 XA_RECOVER 请求的响应。XA TM 将假定数据库已落实此事务。但是，事实上数据处理将丢失，并且回滚了该事务的外观。这将导致发生数据完整性问题，因为 XA TM COMMITTED（落实）了此事务涉及到的所有其他资源。

使用 NEARSYNC 是数据完整性与事务响应时间之间一个很好的折衷，因为在此方式下，主数据库和备用数据库发生崩溃的可能性都比较小。但是，数据库管理员仍然需要了解存在数据完整性问题的可能性。

- ASYNC

  在这三种方式中，此方式在主数据库发生故障的情况下发生事务丢失的可能性最大，但是其事务响应时间最短。仅当主数据库已将日志写入它自己的日志文件，并且已将日志传递至主数据库的主机上的 TCP 层时，主数据库才认为已成功写入日志。主数据库不会等待来自备用数据库的任何种类的确认。当主数据库认为已经落实了相关事务时，有可能仍在将日志写入备用数据库。

  对于 NEARSYNC 中描述的相同情况，该方式丢失事务信息的可能性比 NEARSYNC 方式会更高。因此，发生数据完整性问题的可能性比使用 NEARSYNC 方式时要高，与 SYNC 方式比较起来就更高。

DB2 ESE 分区数据库环境:

在分区数据库环境中使用 ACR 也可能会导致数据完整性问题。如果备用数据库被定义为同一数据库的不同数据库分区，则上面“高可用性灾难恢复”NEARSYNC 部分所描述的方案中的恢复不确定事务可能会导致数据完整性问题。之所以发生数据完整性问题是因为各个数据库分区之间不共享数据库事务日志。因此，备用数据库（数据库分区 B）将不了解主数据库（数据库分区 A）中存在的不确定事务。

DB2 ESE 非分区数据库环境:

在非分区数据库环境中使用 ACR 也可能会导致数据完整性问题。假定没有使用磁盘故障转移技术（例如，“IBM AIX 高可用性集群多处理器”（HACMP™）、“Microsoft 集群服务（MSCS）或 HP 公司的 Service Guard），则当主数据库失败时，备用数据库中将不具有主数据库上存在的数据库事务日志。因此，上面“高可用性灾难恢复”NEARSYNC 部分所描述的方案中的恢复不确定事务可能会导致数据完整性问题。

访问分区数据库的事务

在分区数据库环境中，可将用户数据分布在各数据库分区中。访问该数据库的应用程序与其中一个数据库分区（协调程序节点）连接，并向其发送请求。不同的应用程序可以与不同的数据库分区连接，而同一个应用程序可以为不同的连接选择不同的数据库分区。

对于对分区数据库环境中的某个数据库执行的事务，所有的访问都必须通过*同*一个数据库分区。即，从启动该事务起，直到落实该事务为止（包括落实时的那一刻），都必须使用同一个数据库分区。

对分区数据库执行的任何事务都必须在断开连接之前落实。

相关概念:

- 第 201 页的『X/Open 分布式事务处理模型』
- 『高可用性灾难恢复概述』（《*数据恢复及高可用性指南与参考*》）

相关参考:

- 第 207 页的『xa_open 字符串格式』

xa_open 字符串格式

DB2 数据库 Linux 版、UNIX 版和 Windows 版和 DB2 Connect 版本 8 修订包 3 和更新版本的 xa_open 字符串格式:

以下是 xa_open 字符串的格式:

```
parm_id1 = <parm value>,parm_id2 = <parm value>, ...
```

以何顺序指定这些参数并不重要。下面描述了 *parm_id* 的有效值。

注: 除非明确说明，否则这些参数不区分大小写，并且没有缺省值。

AXLIB
包含 TP 监视器的 **ax_reg** 和 **ax_unreg** 函数的库。DB2 使用此值来获取必需的 **ax_reg** 和 **ax_unreg** 函数的地址。可使用它来覆盖根据 TPM 参数假定的值，它也可以由未出现在 TPM 列表中的 TP 监视器使用。在 AIX 上，如果库是一个归档库，则除了指定库名之外，还应该指定归档成员。例如：`AXLIB=/usr/mqm/lib/libmqmax_r.a(libmqmax_r.o)`。此参数是可选的。

CHAIN_END
xa_end 链接标志。有效值是 `T`、`F` 或没有值。XA_END 链接是 DB2 可用来减少网络流的一种优化措施。若 TP 监视器环境可以保证在调用 **xa_end** 之后，将立即在同一线程或进程中调用 **xa_prepare**，且 CHAIN_END 打开，则 xa_end 标志将与 **xa_prepare** 命令链接，从而消去一个网络流。值 `T` 表示 CHAIN_END 打开；值 `F` 表示 CHAIN_END 关闭；未指定值表示 CHAIN_END 打开。可使用此参数来覆盖从指定的 TPM 值派生的设置。如果未指定此参数，则使用缺省值 `F`。

CREG
xa_start 链接标志。有效值为 `T`、`F` 或没有值。 **xa_start** 链接是 DB2 用来减少网络流的一种优化措施。仅当 TP 监视器正在使用静态注册时该参数才有效（请参阅 SREG）。TP 监视器环境是这样的环境：它可以保证在调用 XA API **xa_start** 之后立即调用 SQL 语句。如果将 CREG 设置为 `T`，则 SQL 语句将被链接至 **xa_start** 请求，从而减少了一个网络流。可使用此参数来覆盖从指定的 TPM 值派生的设置。如果未指定此参数，则使用缺省值 `F`。

CT
连接超时。有效值为 0 - 32767。CT 指定应用程序在尝试与服务器建立连接之前将等待的时间（以秒计）。如果在指定的时间内未建立连接，则将返回一个错误。指定值 0 表示应用程序将尝试等待到建立了连接为止，而无论这要等待多长时间。但是，连接尝试可能被缺省 TCP/IP 超时设置终止。如果未指定此参数，则使用缺省值 `0`。

DB
数据库别名。应用程序用来访问数据库的数据库别名。必须指定此参数。

HOLD_CURSOR
指定是否跨事务落实保持游标。有效值是 `T`、`F` 或没有值。通常，TP 监视器将线程或进程复用于多个应用程序。要确保新装入的应用程序不会继承由先前应用程序打开的游标，游标在落实之后被关闭。如果 HOLD_CURSORS 为打开，则带有挂起属

性的游标不会关闭，并且将在事务落实边界保持不变。当使用此选项时，全局事务必须在同一线程的控制下落实或回滚。如果 HOLD_CURSOR 关闭，则将拒绝打开任何带有挂起属性的游标。值 T 表示 HOLD_CURSOR 打开；值 F 表示 HOLD_CURSOR 关闭；未指定值表示 HOLD_CURSOR 打开。可使用此参数来覆盖从指定的 TPM 值派生的设置。如果未指定此参数，则使用缺省值 F。

PWD

密码。与用户标识相关联的密码。若指定用户标识，则是必需的。此参数区分大小写。

SREG

静态注册。有效值为 T、F 或没有值。DB2 支持用两种方法注册全局事务。第一种方法是“动态注册”，在此方法中，DB2 将调用 TP 的 **ax_reg** 函数来注册事务（请参阅 AXLIB）。第二种方法是“静态注册”，在此方法中，TP 将调用 XA API **xa_start** 来启动全局事务。请注意，动态注册和静态注册是互斥的。如果未指定此参数，则使用缺省值 F。

SUSPEND_CURSOR

指定当暂挂事务控制线程时是否保留游标。有效值是 T、F 或没有值。挂起事务分支的 TP 监视器可以将暂挂的线程或进程复用于其他事务。如果 SUSPEND_CURSOR 关闭，则除了带有挂起属性的游标之外，将关闭所有游标。当恢复暂挂的事务时，应用程序必须再次获取游标。如果 SUSPEND_CURSOR 打开，则不关闭任何打开的游标，并且在恢复暂挂的事务时对该事务是可用的。值 T 表示 SUSPEND_CURSOR 打开；值 F 表示 SUSPEND_CURSOR 关闭；未指定值表示 SUSPEND_CURSOR 打开。可使用此参数来覆盖从指定的 TPM 值派生的设置。如果未指定此参数，则使用缺省值 F。

TOC

所有“DB2 XA 连接”被绑定至的实体“控制线程”。有效值为 T 或 P，或者是未设置。TOC 是在其中绑定所有 DB2 XA 连接的实体。在实体中形成的所有“DB2 XA 连接”都必须是唯一的。也就是说，它们不能与实体中的同一个数据库具有两个连接。TOC 具有两个参数：T（OS 线程）和 P（OS 进程）。当设置为值 T 时，在特定“OS 线程”下形成的所有“DB2 XA 连接”仅对该线程是唯一的。多个线程不能共享“DB2 XA 连接”。每个 OS 线程都必须形成它自己的一组“DB2 XA 连接”。当设置为值 P 时，对于“OS 进程”，所有“DB2 XA 连接”都是唯一的，而 OS 线程之间却可以共享所有“XA 连接”。如果未指定此参数，则使用缺省值 T。

TPM

事务处理监视器名。正在使用的 TP 监视器的名称。有关受支持的值，请参阅下一个表。可指定此参数以允许多个 TP 监视器使用单个 DB2 实例。指定的值将覆盖 *tp_mon_name* 数据库管理器配置参数中指定的值。此参数是可选的。

UID

用户标识。指定有权与数据库连接的用户标识。若指定密码，则是必需的。此参数区分大小写。

UREGNM

用户注册表名。在使用标识映射服务时，此参数指定 UID 参数中指定的用户名所属注册表的名称。

TCTX

指定事务是否应该使用可信连接。有效值是 TRUE 或 FALSE。如果此参数设置为 TRUE，它将指示事务管理器尝试打开可信连接。

TPM 和 tp_mon_name 值:

xa_open 字符串 TPM 参数和 *tp_mon_name* 数据库管理器配置参数用来为 DB2 指示正在使用的 TP 监视器。*tp_mon_name* 值适用于整个 DB2 实例。TPM 参数仅适用于特定 XA 资源管理器。TPM 值覆盖 *tp_mon_name* 参数。TPM 和 *tp_mon_name* 参数的有效值如下所示:

表 40. TPM 和 tp_mon_name 的有效值

TPM 值	TP 监视器产品	内部设置
CICS	IBM TxSeries CICS	AXLIB=libEncServer（对于 Windows） =/usr/lpp/encina/lib/libEncServer （对于基于 UNIX 的系统） HOLD_CURSOR=T CHAIN_END=T SUSPEND_CURSOR=F TOC=T
ENCINA	IBM TxSeries Encina® 监视器	AXLIB=libEncServer（对于 Windows） =/usr/lpp/encina/lib/libEncServer （对于基于 UNIX 的系统） HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F TOC=T
MQ	IBM MQSeries®	AXLIB=mqmax （对于 Windows） =/usr/mqm/lib/libmqmax_r.a (对于 AIX 线程应用程序） =/usr/mqm/lib/libmqmax.a （对于 AIX 非线程应用程序） =/opt/mqm/lib/libmqmax.so （对于 Solaris） =/opt/mqm/lib/libmqmax_r.sl （对于 HP 线程应用程序） =/opt/mqm/lib/libmqmax.sl （对于 HP 非线程应用程序） =/opt/mqm/lib/libmqmax_r.so （对于 Linux 线程应用程序） =/opt/mqm/lib/libmqmax.so （对于 Linux 非线程应用程序） HOLD_CURSOR=F CHAIN_END=F SUSPEND_CURSOR=F TOC=P
CB	IBM Component Broker	AXLIB=somtrx1i （对于 Windows） =libsomtrx1 （对于基于 UNIX 的系统） HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F TOC=T
SF	IBM San Francisco	AXLIB=ibmsfDB2 HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F TOC=T

表 40. *TPM 和 tp_mon_name 的有效值 (续)*

TPM 值	TP 监视器产品	内部设置
TUXEDO	BEA Tuxedo	AXLIB=libtux HOLD_CURSOR=F CHAIN_END=F SUSPEND_CURSOR=F TOC=T
MTS	Microsoft Transaction Server	并不一定要为 MTS 配置 DB2。DB2 的 ODBC 驱动程序会自动检测 MTS。
JTA	Java 事务 API	并不一定要为“企业 Java 服务器”（EJS）（例如，IBM WebSphere）配置 DB2。DB2 的 JDBC 驱动程序自动检测此环境。因此，忽略此 TPM 值。

较早版本的 xa_open 字符串格式:

较早版本的 DB2 使用此处描述的 xa_open 字符串格式。为了保持兼容，仍支持此格式。应在可能的时候将应用程序迁移至新格式。

将每个数据库定义为事务管理器（TM）的一个单独的资源管理器（RM），且必须用 xa_open 字符串标识该数据库，该字符串的语法如下:

```
"database_alias<,userid,password>"
```

database_alias 用于指定数据库的别名。除非您在创建数据库之后显式地编目了一个别名，否则，别名与数据库名称相同。用户名和密码是可选的（这取决于认证方法），它们将用于向数据库提供认证信息。

示例:

1. 您正在 Windows 上使用 IBM TxSeries CICS。TxSeries 文档指示您需要使用值 `libEncServer:C` 来配置 *tp_mon_name*。这仍然是一种可以接受的格式；然而，对于 DB2 数据库 Linux 版、UNIX 版和 Windows 版或 DB2 Connect 版本 8 修订包 3 和更新版本，可以选择:
   - 指定值为 CICS 的 *tp_mon_name*（建议用于此方案）:

     ```
     db2 update dbm cfg using tp_mon_name CICS
     ```

     对于向 CICS 定义的每个数据库，在“区域”—>“资源”—>“产品”—> XAD —>“资源管理器初始化字符串”中，指定:

     ```
     db=dbalias,uid=userid,pwd=password
     ```
   - 对于向 CICS 定义的每个数据库，在“区域”—>“资源”—>“产品”—> XAD —>“资源管理器初始化字符串”中，指定:

     ```
     db=dbalias,uid=userid,pwd=password,tpm=cics
     ```
2. 您正在 Windows 上使用 IBM MQSeries。MQSeries 文档指示您需要使用值 mqmax 来配置 *tp_mon_name*。这仍然是一种可以接受的格式；然而，对于 DB2 数据库 Linux 版、UNIX 版和 Windows 版或 DB2 Connect 版本 8 修订包 3 和更新版本，可以选择:
   - 指定值为 MQ 的 *tp_mon_name*（建议用于此方案）:

     ```
     db2 update dbm cfg using tp_mon_name MQ
     ```

对于向 CICS 定义的每个数据库，在“区域”—>“资源”—>“产品”—> XAD —>“资源管理器初始化字符串”中，指定:

uid=*userid*,db=*dbalias*,pwd=*password*

- 对于向 CICS 定义的每个数据库，在“区域”—>“资源”—>“产品”—> XAD —>“资源管理器初始化字符串”中，指定:

  uid=*userid*,db=*dbalias*,pwd=*password*,tpm=mq

3. 您正在 Windows 上同时使用 IBM TxSeries CICS 和 IBM MQSeries。正在使用单个 DB2 实例。在此方案中，应按如下方法进行配置:

   a. 对于向 CICS 定义的每个数据库，在“区域”—>“资源”—>“产品”—> XAD —>“资源管理器初始化字符串”中，指定:

      pwd=*password*,uid=*userid*,tpm=cics,db=*dbalias*

   b. 对于在队列管理器属性中定义成资源的每个数据库，将 XaOpenString 指定为:

      db=*dbalias*,uid=*userid*,pwd=*password*,tpm=mq

4. 您正在 Windows 上开发您自己的符合 XA 的事务管理器（XA TM），并且您想告知 DB2“myaxlib”库具有必需的函数 **ax_reg** 和 **ax_unreg**。“myaxlib”库在 PATH 语句中指定的目录中。您可以选择:

   - 指定值为 myaxlib 的 *tp_mon_name*:

     ```
     db2 update dbm cfg using tp_mon_name myaxlib
     ```

     并且，对于向 XA TM 定义的每个数据库，指定 xa_open 字符串:

     db=*dbalias*,uid=*userid*,pwd=*password*

   - 对于向 XA TM 定义的每个数据库，指定 xa_open 字符串:

     db=*dbalias*,uid=*userid*,pwd=*password*,axlib=myaxlib

5. 您正在 Windows 上开发您自己的符合 XA 的事务管理器（XA TM），并且您想告知 DB2“myaxlib”库具有必需的函数 **ax_reg** 和 **ax_unreg**。“myaxlib”库在 PATH 语句中指定的目录中。您还想启用 XA END 链接。您可以选择:

   - 对于向 XA TM 定义的每个数据库，指定 xa_open 字符串:

     db=*dbalias*,uid=*userid*,pwd=*password*,axlib=myaxlib,chain_end=T

   - 对于向 XA TM 定义的每个数据库，指定 xa_open 字符串:

     db=*dbalias*,uid=*userid*,pwd=*password*,axlib=myaxlib,chain_end

相关概念:

- 第 201 页的『X/Open 分布式事务处理模型』

相关参考:

- 『tp_mon_name － 事务处理器监视器名称配置参数』（*《性能指南》*）

使用符合 XA 的事务管理器更新主机或 iSeries 数据库服务器

主机和 iSeries 数据库服务器可能是可更新的，这取决于“XA 事务管理器”的体系结构。

过程:

要支持来自不同进程的落实序列，必须启用 DB2 Connect 连接集中器。要启用 DB2 Connect 连接集中器，将数据库管理器配置参数 *max_connections* 设置为大于

*max_coordagents*的值。注意，DB2 Connect 连接集中器需要 DB2 通用数据库（DB2 UDB）版本 7.1 客户机或更新版本才能支持来自不同进行的 XA 落实序列。

还将需要配置了 DB2 同步点管理器（SPM）的 DB2 Connect。

相关参考:

- 『maxagents － 最大代理程序数配置参数』（*《性能指南》*）
- 『max_connections － 最大客户机连接数配置参数』（*《性能指南》*）

手工解决不确定事务

符合 XA 的事务管理器（事务处理监视器）使用类似于 DB2 事务管理器使用的两阶段落实过程。两个环境之间的主要区别是，TP 监视器提供日志记录和控制该事务的功能，而不是 DB2 事务管理器和事务管理器数据库来提供此功能。

当使用符合 XA 的事务管理器时，可能发生类似于 DB2 事务管理器发生的错误。类似于 DB2 事务管理器，符合 XA 的事务管理器将试图使不确定事务再同步。

若您不能等待事务管理器自动解决不确定事务，则您可以手工解决它们。此手工过程有时称为"作启发式决策"。

LIST INDOUBT TRANSACTIONS 命令（使用 WITH PROMPTING 选项）或相关的一组 API（*db2XaListIndTrans*、*sqlxphcm*、*sqlxhfrg* 和 *sqlxphrl*）允许您查询、落实和回滚不确定事务。另外，它还允许通过除去日志记录并释放日志空间来"忘记"已经以试探方式落实或回滚的事务。

限制:

*极为谨慎*地使用这些命令（或相关的 API）来手工解决不确定事务，并且只有在万不得已的情况下才使用这些命令。最好的策略是等待事务管理器驱动再同步过程。若您在其中一个参与数据库中以手工方式落实或回滚一个事务，并对另一个参与的数据库执行相反的操作，可能会遇到数据完整性问题。校正数据完整性问题要求您了解该应用程序的逻辑，标识已更改或回滚的数据，然后执行数据库的时间点恢复，或以手工方式撤销或重新执行更改。

若您不能等待事务管理器启动再同步过程，且您必须释放不确定事务占用的资源，则必须执行试探性操作。若事务管理器无法在延长期内执行再同步，且该不确定事务占用着急需的资源，则可能发生这种情况。不确定事务占用着事务管理器或资源管理器成为不可用之前与此事务相关的资源。对于数据库管理器，这些资源包括对该事务占用的表和索引、日志空间和存储器的锁定。每个不确定事务还会递减（减一）数据库可以处理的并发事务的最大数目。而且，除非解析了所有不确定事务，否则不能进行脱机备份。

在下列情况下，需要试探性忘记功能:

- 当以试探方式落实或回滚事务导致日志满情况时（此情况在 LIST INDOUBT TRANSACTIONS 命令的输出中指示）
- 当要执行脱机备份时

试探性忘记功能释放由不确定事务占用的日志空间。隐含情况为，若一个事务管理器最终对此不确定事务执行了再同步操作，则可能会作出错误的决定来落实或回滚其他资源管理器，因为在此资源管理器中没有该事务的日志记录。通常，“丢失”日志记录意味着资源管理器已回滚该事务。

过程:

要手工解决不确定事务:

1. 与您要完成其所有事务的数据库连接。
2. 显示不确定事务:
   - 对于 DB2 数据库服务器，使用 LIST INDOUBT TRANSACTIONS WITH PROMPTING 命令。*xid* 表示全局事务标识，它与事务管理器和参与事务的其他资源管理器使用的 *xid* 完全相同。
   - 对于主机或 iSeries 数据库服务器，可以使用下列其中一种方法:
     - 可以直接从主机或 iSeries 服务器获取不确定信息。

       要直接从 DB2 z/OS 和 OS/390 版获取不确定信息，调用 DISPLAY THREAD TYPE(INDOUBT) 命令。使用 RECOVER 命令来作出启发式决策。要直接从 DB2 iSeries 版获取不确定信息，调用 **wrkcmtdfn** 命令。
     - 可以从用来访问主机或 iSeries 数据库服务器的 DB2 Connect 服务器获取不确定信息。

       要从 DB2 Connect 服务器获取不确定信息，首先通过连接由数据库管理器配置参数 *spm_name* 的值所表示的 DB2 实例来与 DB2 同步点管理器连接。然后发出 LIST DRDA INDOUBT TRANSACTIONS WITH PROMPTING 命令来显示不确定事务并作出启发式决策。另外，可以从客户机应用程序中调用 *sqlcspqy* API 来列示 DRDA® 不确定事务。
3. 对于已列示或显示的每个不确定事务，使用显示的关于应用程序和操作环境的信息来确定其他参与的资源管理器。
4. 确定要对每个不确定事务执行的操作:
   - 如果该事务管理器是可用的，且资源管理器中的不确定事务是由于在第二个落实阶段期间或更早的再同步过程期间资源管理器不可用导致的，则您应该执行下列操作:
     a. 检查该事务管理器的日志，以确定对其他资源管理器执行过什么操作。
     b. 对该数据库执行相同的操作；即使用 LIST INDOUBT TRANSACTIONS WITH PROMPTING 命令试探性落实或试探性回滚该事务。
   - 如果事务管理器不可用，则应使用其他参与资源管理器中该事务的状态，以确定您应该执行什么操作:
     - 如果其他资源管理器中至少有一个已经落实该事务，则应在所有资源管理器中试探性落实该事务。
     - 如果其他资源管理器中至少有一个已经回滚该事务，则应试探性回滚该事务。
     - 如果该事务在参与的所有资源管理器中都处于“已准备”（不确定）状态，则应试探性回滚该事务。
     - 如果其他资源管理器中有一个或多个不可用，则应试探性回滚该事务。

要从运行于 UNIX 或 Windows 操作系统上的 DB2 中获取不确定事务信息，请连接至数据库并发出 LIST INDOUBT TRANSACTIONS WITH PROMPTING 命令，或者从客户机应用程序中调用 *db2XaListIndTrans* API。

相关概念:

- 第 214 页的『不确定事务管理 API』
- 第 195 页的『两阶段落实』

相关参考:

- 『LIST DRDA INDOUBT TRANSACTIONS command』（*Command Reference*）
- 『LIST INDOUBT TRANSACTIONS command』（*Command Reference*）
- 『db2XaListIndTrans API - List indoubt transactions』（*Administrative API Reference*）
- 『sqlcspqy API - List DRDA indoubt transactions』（*Administrative API Reference*）
- 『sqlxhfrg API - Forget transaction status』（*Administrative API Reference*）
- 『sqlxphcm API - Commit an indoubt transaction』（*Administrative API Reference*）
- 『sqlxphrl API - Roll back an indoubt transaction』（*Administrative API Reference*）

相关样本:

- 『dbxamon.c -- Show and roll back indoubt transactions.』

不确定事务管理 API

可以在分布式事务处理（DTP）环境中使用数据库。

为工具编写者提供了一组 API，以便在资源所有者（例如，数据库管理员）不能等到“事务管理器”（TM）来执行*再同步*操作时使用这些 API 对不确定事务执行试探函数。例如，当通信线路断开，并且不确定事务正在占用所需资源时，就可能出现此情况。对于数据库管理器，这些资源包括对该事务使用的表和索引、日志空间和存储器的锁定。每个不确定事务还会递减（减一）数据库管理器可以处理的并发事务的最大数目。

试探 API 具有查询、落实和回滚不确定事务的能力，并且具有通过除去日志记录并释放日志页来取消已通过试探方式落实或回滚的事务的能力。

警告: 应小心使用试探 API，并且只有在万不得已的情况下才使用它。TM 应驱动再同步事件。如果 TM 具有启动再同步操作的操作员命令，则应该使用该命令。如果用户不能等到 TM 启动的再同步，则试探操作是必需的。

虽然没有一套方法来执行这些操作，但下列准则可能会有用:

- 使用 db2XaListIndTrans 函数显示不确定事务。这些不确定事务具有状态“P”（已准备），并且未连接。*xid* 的 *gtrid* 部分是全局事务标识，该标识与参与全局事务的其他资源管理器（RM）中的全局事务标识相同。
- 使用应用程序和操作环境知识来标识其他参与的 RM。
- 如果事务管理器是 CICS，并且唯一的 RM 是 CICS 资源，则执行试探回滚。
- 如果事务管理器不是 CICS，则使用它来确定与不确定事务具有相同 *gtrid* 的事务的状态。
- 如果至少落实或回滚了一个 RM，则执行试探落实或回滚。

- 如果它们都处于已准备状态，则执行试探回滚。
- 如果至少一个 RM 不可用，则执行试探回滚。

如果事务管理器可用，且不确定事务是由于在第二个阶段或在更早的再同步过程中 RM 不可用导致的，则 DBA 应根据 TM 的日志确定对其他 RM 执行过什么操作，然后再执行相同的操作。*gtrid* 是 TM 与 RM 之间的匹配键。

除非由于通过试探方式落实或回滚事务导致日志已满，否则不要执行 sqlxhfrg。忘记功能释放此不确定事务占用的日志空间。如果事务管理器最终对此不确定事务执行再同步操作，则 TM 可能会作出错误的决定来落实或回滚其他 RM，因为在此 RM 中找不到记录。通常，缺少记录暗示 RM 已回滚。

相关参考:

- 『db2XaListIndTrans API - List indoubt transactions』（*Administrative API Reference*）
- 『sqlcspqy API - List DRDA indoubt transactions』（*Administrative API Reference*）
- 『sqlxhfrg API - Forget transaction status』（*Administrative API Reference*）
- 『sqlxphcm API - Commit an indoubt transaction』（*Administrative API Reference*）
- 『sqlxphrl API - Roll back an indoubt transaction』（*Administrative API Reference*）

XA 事务管理器的安全性注意事项

TP 监视器预分配一组服务器进程，并由在这些服务器进程标识下的不同用户运行事务。对于该数据库，每个服务器进程表现为一个有许多工作单元的大应用程序，所有工作单元都在与该服务器进程相关的相同标识下运行。

例如，在使用 CICS 的 AIX 环境中，当启动 TXSeries® CICS 区域时，该区域与用于定义它的 AIX 用户名相关联。所有 CICS 应用程序服务器进程也都在此 TXSeries CICS 主标识（通常定义为“cics”）下运行。CICS 用户可以在他们的 DCE 登录标识下调用 CICS 事务，且在 CICS 中时，他们还可以使用 CESN 注册事务更改他们的标识。在任何一种情况中，RM 都无法使用最终用户的标识。因此， CICS 应用程序进程可能正在代表许多用户运行事务，但是，对于 RM 来说，它就像是一个带有许多工作单元的单一程序，而这些工作单元具有相同的“cics”标识。您可以选择在 xa_open 字符串中指定用户标识和密码，并将使用该用户标识来代替“cics”标识以与数据库连接。

对于静态 SQL 语句，没有很大影响，因为是使用绑定程序的特权而非最终用户的特权来访问数据库。但是，这意味着必须将数据库程序包的 EXECUTE 特权授予服务器标识，而不是最终用户标识。

对于动态语句，它们在运行时执行访问认证，必须将数据库对象的访问特权授予服务器标识，而非那些对象的实际用户。此时必须依靠 TP 监视器系统来确定允许用户运行的程序，而不是依靠数据库来控制特定用户的访问。必须授予服务器标识它的 SQL 用户需要的所有特权。

要确定谁访问了数据库表或视图，您可以执行下列步骤:

1. 从 SYSCAT.PACKAGEDEP 目录视图，获取取决于该表或视图的所有程序包的列表。
2. 通过安装期间使用的名称约定，确定与这些程序包对应的服务器程序（例如，CICS 程序）的名称。

3. 确定可以调用这些程序的客户机程序（例如，CICS 事务），然后使用 TP 监视器日志（例如，CICS 日志）来确定谁运行了这些事务或程序以及是何时运行的。

相关概念:

- 第 201 页的『X/Open 分布式事务处理模型』

XA 事务管理器的配置注意事项

当您设置 TP 监视器环境时，应该考虑下列配置参数:

- *tp_mon_name*

  此数据库管理器配置参数标识要使用的 TP 监视器产品的名称（例如，“CICS”或“ENCINA”）。

- *tpname*

  此数据库管理器配置参数标识远程事务程序的名称，当数据库客户机使用 APPC 通信协议向数据库服务器发出分配请求时，必须使用该事务程序。此值是在服务器的配置文件中设置的，且必须与在 SNA 事务程序中配置的事务处理器（TP）的名称相同。

- *tm_database*

  因为 DB2 数据库 Linux 版、UNIX 版和 Windows 版不会协调 XA 环境中的事务，所以不对协调 XA的事务使用此数据库管理器配置参数。

- *maxappls*

  此数据库配置参数指定允许的活动应用程序的最大数目。此参数的值必须等于或大于连接的应用程序数，加上可能在完成一个两阶段落实或回滚的过程中并存的这些应用程序的数目之和。此和随后应加上在任何时刻可能存在的预期不确定事务数。

  对于 TP 监视器环境（例如，TXSeries CICS），可能需要增大 *maxappls* 参数的值。这有助于确保使所有 TP 监视器进程都适应。

- *autorestart*

  此数据库配置参数指定在需要时是否自动调用 RESTART DATABASE 例程。缺省值是 `YES`（即，启用）。

  包含不确定事务的数据库需要重新启动数据库操作才能启动。若断开与该数据库的上一个连接时未启用 *autorestart*，则下一个连接将失败并需要显式的 RESTART DATABASE 调用。这种情况将一直存在，直到事务管理器的再同步操作或管理员启动的试探性操作除去了不确定事务为止。当发出 RESTART DATABASE 命令时，若该数据库中有任何不确定事务，将返回一条消息。管理员就可以使用 LIST INDOUBT TRANSACTIONS 命令和其他的命令行处理器命令来获取有关那些不确定事务的信息。

相关概念:

- 第 201 页的『X/Open 分布式事务处理模型』

相关参考:

- 『tpname － APPC 事务程序名配置参数』（《*性能指南*》）
- 『autorestart － 启用自动重新启动配置参数』（《*性能指南*》）

- 『LIST INDOUBT TRANSACTIONS command』（*Command Reference*）
- 『maxappls – 最大活动应用程序数配置参数』（《性能指南》）
- 『RESTART DATABASE command』（*Command Reference*）
- 『tm_database – 事务管理器数据库名称配置参数』（《性能指南》）
- 『tp_mon_name – 事务处理器监视器名称配置参数』（《性能指南》）

DB2 数据库 Linux 版、UNIX 版和 Windows 版支持的 XA 功能

DB2 数据库 Linux 版、UNIX 版和 Windows 版支持 *X/Open CAE 规范分布式事务处理: XA 规范*中定义的 XA91 规范，但是下列情况除外:

- 异步服务

  XA 规范允许该接口使用异步服务，这样可以在稍后的某个时间检查请求的结果。数据库管理器要求以同步方式调用请求。
- 注册

  XA 接口允许以两种方式来注册 RM: 静态注册和动态注册。DB2 既支持动态注册也支持静态注册。DB2 提供了两个开关来控制使用的注册类型。
  - *db2xa_switch_std* 用于动态注册
  - *db2xa_switch_static_std* 用于静态注册
- 关联迁移

  DB2 V9.1 不支持控制线程之间的事务迁移。

XA 开关用法和位置

按照 XA 接口的要求，数据库管理器提供了 xa_switch_t 类型的 *db2xa_switch_std* 和 *db2xa_switch_static_std* 外部 C 变量来将 XA 开关结构返回给 TM。不是返回各种 XA 函数的地址，而是返回下列字段:

字段 **值**

名称
数据库管理器的产品名。例如，IBM DB2 版本 9.1 AIX 版。

标志
对于 *db2xa_switch_std*，设置 TMREGISTER | TMNOMIGRATE

显式声明 DB2 V9.1 使用动态注册，并且 TM 不应该使用关联迁移。隐式声明不支持异步操作。

对于 *db2xa_switch_static_std*，设置 TMNOMIGRATE

显式声明 DB2 V9.1 使用静态注册，并且 TM 不应该使用关联迁移。隐式声明不支持异步操作。

版本
必须为零。

使用 DB2 数据库 Linux 版、UNIX 版和 Windows 版 XA 开关

XA 体系结构要求“资源管理器”（RM）提供*开关*，该开关使“XA 事务管理器”（TM）可以访问 RM 的 **xa_** 例程。RM 开关使用称为 xa_switch_t 的结构。该开关包含 RM 的名称、指向 RM 的 XA 入口点的非 NULL 指针、标志和版本号。

Linux 和 UNIX

可以通过下面两种方法之一来获得 DB2 数据库 Linux 版、UNIX 版和 Windows 版的开关:

- 通过更高一级的间接方法。在 C 程序中，可以在使用 *db2xa_switch_std* 或 *db2xa_switch_static_std* 之前定义下列宏来实现:

```
#define db2xa_switch_std (*db2xa_switch_std)
#define db2xa_switch_static_std (*db2xa_switch_std)
```

- 通过调用 **db2xacic_std** 或 **db2xacicst_std**

  DB2 提供了这些 API，它们将返回 *db2xa_switch_std* 或 *db2xa_switch_static_std* 结构的地址。此函数的原型为:

```
struct xa_switch_t * SQL_API_FN  db2xacic_std( )
struct xa_switch_t * SQL_API_FN  db2xacicst_std( )
```

不管使用哪种方法，必须将应用程序与 `libdb2` 链接。

Windows

指向 *xa_switch* 结构、*db2xa_switch_std* 或 *db2xa_switch_static_std* 的指针是作为 DLL 数据导出的。这意味着使用此结构的 Windows 应用程序必须以下列三种方法之一来引用它:

- 通过更高一级的间接方法。在 C 程序中，可以在使用 *db2xa_switch_std* 或 *db2xa_switch_static_std* 之前定义下列宏来实现:

```
#define db2xa_switch_std (*db2xa_switch_std)
#define db2xa_switch_static_std (*db2xa_switch_std)
```

- 如果使用 Microsoft Visual C++ 编译器，则可以将 *db2xa_switch_std* 或 *db2xa_switch_static_std* 定义为:

```
extern __declspec(dllimport) struct xa_switch_t db2xa_switch_std
extern __declspec(dllimport) struct xa_switch_t db2xa_switch_static_std
```

- 通过调用 **db2xacic_std** 或 **db2xacicst_std**

  DB2 提供了此 API，它将返回 *db2xa_switch_std* 或 *db2xa_switch_static_std* 结构的地址。此函数的原型为:

```
struct xa_switch_t * SQL_API_FN  db2xacic_std( )
struct xa_switch_t * SQL_API_FN  db2xacicst_std( )
```

使用这些方法中任何一种，必须将应用程序与 `db2api.lib` 链接。

C 代码示例

以下代码演示在任何 DB2 V9.1 平台上通过 C 程序访问 *db2xa_switch_std* 或 *db2xa_switch_static_std* 的不同方法。务必将应用程序与适当的库链接。

```
#include <stdio.h>
#include <xa.h>

struct xa_switch_t * SQL_API_FN  db2xacic_std( );

#ifdef DECLSPEC_DEFN
extern __declspec(dllimport) struct xa_switch_t db2xa_switch_std;
#else
#define db2xa_switch_std (*db2xa_switch_std)
extern struct xa_switch_t db2xa_switch_std;
#endif
```

```
main( )
   {
      struct xa_switch_t *foo;
      printf ( "%s \n", db2xa_switch_std.name );
      foo = db2xacic_std();
      printf ( "%s \n", foo->name );
      return ;
   }
```

相关概念:

- 第 201 页的『X/Open 分布式事务处理模型』

XA 接口问题确定

若从 TM 发出 XA 请求期间检测到错误，则应用程序可能无法从 TM 获取错误代码。若您的程序异常终止，或从 TP 监视器或 TM 获得加密的返回码，则应该检查“首次故障服务日志”，当诊断级别 3 或更高级别生效时，它将报告 XA 错误信息。

您还应该参考控制台消息、TM 错误文件或关于所用的外部事务处理软件的其他产品特定消息。

数据库管理器将所有特定于 XA 的错误写入“首次故障服务日志”，并提供 SQLCODE -998（事务或试探性错误）和适当的原因码。以下是一些更为常见的错误:

- xa_open 字符串中的语法无效。
- 由于下列原因之一，未能与打开字符串中指定的数据库连接:
  - 数据库未编目。
  - 数据库未启动。
  - 服务器应用程序的用户名或密码无权与该数据库连接。
- 通信错误。

相关概念:

- 第 201 页的『X/Open 分布式事务处理模型』

相关参考:

- 第 207 页的『xa_open 字符串格式』

XA 事务管理器配置

配置 IBM WebSphere Application Server

IBM WebSphere Application Server 是基于 Java 的应用程序服务器。它可以通 DB2 JDBC 驱动程序提供的“Java 事务 API”（JTA）来使用 DB2 数据库 Linux 版、UNIX 版和 Windows 版 XA 支持。请参阅有关如何将“Java 事务 API”与 WebSphere Application Server 配合使用的 IBM WebSphere 文档。WebSphere Application Server 文档可通过访问以下网址来联机查看: http://www.ibm.com/software/webservers/appserv/infocenter.html。

配置 IBM TXSeries CICS

有关如何配置 IBM TXSeries CICS 以将 DB2 数据库 Linux 版、UNIX 版和 Windows 版用作资源管理器的信息，请参阅 *IBM TXSeries CICS Administration Guide*。TXSeries 文档可通过访问如下站点联机查看：http://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/。

主机和 iSeries 数据库服务器可以参与协调 CICS 事务。

配置 IBM TXSeries Encina

以下是通过 DB2 Connect 访问时，将 Encina 监视器与 DB2 数据库 Linux 版、UNIX 版和 Windows 版服务器或 DB2 z/OS 和 OS/390 版、DB2 iSeries 版或 DB2 VSE 和 VM 版集成时需要的各种 API 和配置参数。TXSeries 文档可通过访问以下站点联机查看：http://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/。

主机和 iSeries 数据库服务器可以参与协调 Encina 事务。

配置 DB2 数据库 Linux 版、UNIX 版和 Windows 版

要配置 DB2 数据库 Linux 版、UNIX 版和 Windows 版：

1. 必须在 DB2 数据库目录中定义每个数据库名称。若该数据库是远程数据库，则还必须定义节点目录项。您可以使用“配置助手”或 DB2 命令行处理器（CLP）来执行配置。例如：

   ```
      DB2 CATALOG DATABASE inventdb AS inventdb AT NODE host1 AUTH SERVER
   DB2 CATALOG TCPIP NODE host1 REMOTE hostname1 SERVER svcname1
   ```

2. 如果 DB2 客户机知道它正在处理的是 Encina，则它可以针对 Encina 优化其内部处理。您可以将 *tp_mon_name* 数据库管理器配置参数设置为 ENCINA 来指定此选项。缺省行为是不进行特殊的优化。若设置了 *tp_mon_name*，则该应用程序必须确保执行该工作单元的线程在结束之后也立即落实该工作。不能启动任何其他工作单元。若您的环境不是这样的，则确保 *tp_mon_name* 值是 NONE（或者，通过 CLP 将此值设置为 NULL）。可以通过“控制中心”或 CLP 更新此参数。CLP 命令是：

   ```
   db2 update dbm cfg using tp_mon_name ENCINA
   ```

为每个资源管理器配置 Encina

要为每个资源管理器（RM）配置 Encina，管理员为作为资源管理器的每个 DB2 数据库定义“打开字符串”、“关闭字符串”和“控制线程协议”，然后才能为应用程序中的事务注册资源管理器。可以使用 Enconcole 全屏幕界面或 Encina 命令行界面来执行该配置。例如：

```
monadmin create rm inventdb -open "db=inventdb,uid=user1,pwd=password1"
```

每个 DB2 数据库都具有一个资源管理器配置，且每个资源管理器配置都必须具有 rm 名称（“逻辑 RM 名称”）。为了简化情况，您应该使它与数据库名称相同。

xa_open 字符串包含建立与数据库的连接所需的信息。该字符串的内容是 RM 所独有的。DB2 的 xa_open 字符串包含要打开的数据库的别名以及（可选）与该连接关联的用户标识和密码。注意，此处定义的数据库名称也必须在所有数据库访问都必需的规则数据库目录中编目。

DB2 不使用 xa_close 字符串。

“控制线程协议”确定一个应用程序的代理程序线程是否一次可以处理多个事务。

如果您要访问 DB2 z/OS 和 OS/390 版、DB2 iSeries 版或 DB2 VSE 和 VM 版，则必须使用 DB2 同步点管理器。

从 Encina 应用程序引用 DB2 数据库 Linux 版、UNIX 版和 Windows 版数据库

要从 Encina 应用程序引用 DB2 数据库 Linux 版、UNIX 版和 Windows 版数据库:

1. 使用 Encina 调度策略 API 来指定可在单个 TP 监视器应用程序进程中运行的应用程序代理程序的个数。例如:

   ```
   rc = mon_SetSchedulingPolicy (MON_EXCLUSIVE)
   ```

2. 使用 Encina RM 注册 API 来提供 XA 开关和逻辑 RM 名称，以供 Encina 在应用程序进程中引用 RM 时使用。例如:

   ```
   rc = mon_RegisterRmi ( &db2xa_switch,   /* xa switch */
                          "inventdb",      /* logical RM name */
                          &rmiId );        /* internal RM ID */
   ```

   XA 开关包含 RM 中 TM 可以调用的 XA 例程的地址，它还指定由 RM 提供的功能。DB2 V9.1 的 XA 开关是 db2xa_switch，它位于 DB2 客户机库（在 Windows 操作系统上是 db2app.dll，在基于 UNIX 的系统上是 libdb2）中。

   逻辑 RM 名称是 Encina 使用的名称，但并不是在 Encina 下运行的 SQL 应用程序使用的实际数据库名称。实际数据库名称是在 Encina RM 注册 API 的 xa_open 字符串中指定的。逻辑 RM 名设置为与本示例中的数据库名称相同。

   第三个参数返回 TM 引用此连接所用的内部标识或句柄。

相关概念:

- 『DB2 Connect 和事务处理监视器』（*《DB2 Connect 用户指南》*）

相关参考:

- 『tp_mon_name － 事务处理器监视器名称配置参数』（*《性能指南》*）
- 第 207 页的『xa_open 字符串格式』

配置 BEA Tuxedo

以下是对配置 BEA Tuxedo 以供 DB2 数据库 Linux 版、UNIX 版和 Windows 版使用的过程的描述。记录了 Tuxedo 在使用 DB2 数据库 Linux 版、UNIX 版和 Windows 版的 64 位实例和使用 DB2 数据库 Linux 版、UNIX 版和 Windows 版的 32 位实例之间的一些区别。

注: XA 开关数据结构有新的名称: *db2xa_switch_std* 和 *db2xa_switch_static_std*。API 也有新的名称: *db2xacic* 和 *db2xacicst*。只有在使用 DB2 数据库 Linux 版、UNIX 版和 Windows 版的 32 位实例时，才能使用旧的开关数据结构和 API 名称。

过程:

要配置 Tuxedo 以将 DB2 数据库 Linux 版、UNIX 版和 Windows 版用作资源管理器，执行下列步骤:

1. 按该产品的文档中指定的步骤，安装 Tuxedo。确保执行所有基本的 Tuxedo 配置，包括日志文件和环境变量。

   您还需要一个编译器和“DB2 应用程序开发客户机”。需要时安装它们。
2. 在 Tuxedo 服务器标识中，设置 DB2INSTANCE 环境变量，以引用包含您希望 Tuxedo 使用的数据库的实例。将 PATH 变量设置为包括 DB2 程序目录。确认 Tuxedo 服务器标识可以与 DB2 数据库连接。
3. 使用值 TUXEDO 来更新 *tp_mon_name* 数据库管理器配置参数。
4. 将 DB2 V9.1 的定义添加至 Tuxedo 资源管理器定义文件。在下面的示例中，UDB_XA 是为 DB2 V9.1 在本地定义的 Tuxedo 资源管理器名，而 *db2xa_switch_std* 是类型为 xa_switch_t 的一个结构的 DB2 定义的名称：
   - 对于 AIX。在文件 ${TUXDIR}/udataobj/RM 中，添加定义：
     ```
        # DB2 UDB
     UDB_XA:db2xa_switch_std:-L${DB2DIR} /lib -ldb2
     ```
     其中 {TUXDIR} 是 Tuxedo 的安装目录，而 {DB2DIR} 是 DB2 实例目录。
   - 对于 Windows。在文件 %TUXDIR%\udataobj\rm 中，添加定义：
     ```
        # DB2 UDB
     UDB_XA;db2xa_switch_std;%DB2DIR%\lib\db2api.lib
     ```
     其中 %TUXDIR% 是 Tuxedo 的安装目录，而 %DB2DIR% 是 DB2 实例目录。
5. 为 DB2 构建 Tuxedo 事务监视器服务器程序：
   - 对于 AIX：
     ```
        ${TUXDIR}/bin/buildtms -r UDB_XA -o ${TUXDIR}/bin/TMS_UDB
     ```
     其中，{TUXDIR} 是安装了 Tuxedo 的目录。
   - 对于 Windows：
     ```
     %TUXDIR%\bin\buildtms -r UDB_XA -o %TUXDIR%\bin\TMS_UDB
     ```
6. 构建应用程序服务器。在下面的示例中，-r 选项指定资源管理器名，-f 选项（使用了一次或多次）指定包含应用程序服务的文件，-s 选项指定此服务器的应用程序服务名称，而 -o 选项指定输出服务器文件名：
   - 对于 AIX：
     ```
     ${TUXDIR}/bin/buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
        -o UDBserver
     ```
     其中，{TUXDIR} 是安装了 Tuxedo 的目录。
   - 对于 Windows：
     ```
     %TUXDIR%\bin\buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
        -o UDBserver
     ```
     其中 %TUXDIR% 是安装了 Tuxedo 的目录。
7. 设置 Tuxedo 配置文件以引用 DB2 服务器。在 UDBCONFIG 文件的 *GROUPS 小节中，添加类似如下的条目：
   ```
    UDB_GRP   LMID=simp GRPNO=3
   TMSNAME=TMS_UDB TMSCOUNT=2
   OPENINFO="UDB_XA:db=sample,uid=db2_user,pwd=db2_user_pwd"
   ```

其中，TMSNAME 参数指定您先前构建的事务监视器的服务器程序，而 OPENINFO 参数指定资源管理器名。其后是数据库名称、DB2 数据库用户标识和密码（用于认证）。

您先前构建的应用程序服务器在 Tuxedo 配置文件的 *SERVERS 一节中被引用。

8. 若应用程序正在访问位于 DB2 z/OS 和 OS/390 版、DB2 iSeries 版或者 DB2 VM 和 VSE 版上的数据，则将需要 DB2 Connect XA 集中器。
9. 启动 Tuxedo:

```
tmboot -y
```

在该命令完成之后，Tuxedo 消息应指示服务器已启动。另外，如果您发出 DB2 命令 LIST APPLICATIONS ALL，您应该看到两个连接（在这种情况下），它们由 Tuxedo 配置文件 `UDBCONFIG` 中的 UDB 组的 TMSCOUNT 参数指定。

相关概念:

- 『DB2 Connect 和事务处理监视器』（*《DB2 Connect 用户指南》*）

相关参考:

- 『LIST APPLICATIONS command』（*Command Reference*）
- 『tp_mon_name – 事务处理器监视器名称配置参数』（*《性能指南》*）

第 3 部分 附录

附录 A. 发行版之间的不兼容性

本节说明 DB2 版本 9 与 DB2 通用数据库的前发行版之间的不兼容性。

*不兼容性*是指 DB2 数据库与其前发行版的工作方式不同的部分。若在现有应用程序中使用，将产生预料不到的结果，要求更改应用程序，或降低性能。在本上下文中，“应用程序”是指:

- 应用程序代码
- 第三方实用程序
- 交互式 SQL 查询
- 命令或 API 调用

DB2 通用数据库版本 8 和 DB2 版本 9 之间的不兼容性。它们根据下列类别进行分组:

- 系统目录信息
- 应用程序编程
- SQL
- 数据库安全性和调整
- 实用程序和工具
- 连接和共存
- 消息
- 配置参数

每个不兼容小节都包括对不兼容性的描述、不兼容性的症状或造成的影响以及可能的解决方案。 每个不兼容性描述的开头都有一个指示符，标识不兼容性适用的操作系统:

Windows
DB2 数据库支持的 Microsoft Windows® 平台

UNIX DB2 数据库支持的基于 UNIX® 的平台

不推荐使用和不继续使用的功能

本节描述了当前和将来不推荐使用和不继续使用的功能。此外，本节还描述了 DB2 数据库系统的用户在编写新应用程序或修改现有应用程序时应该记住的计划不兼容性。了解这些更改将有助于您进行当前应用程序开发和将来计划转为使用更新版本的 DB2。

例如，在参考资料中，您将在命令或 SQL 语句语法和描述中发现新参数，并且会说明新参数要取代另一个参数。但是，在一段时间内将继续识别较旧的参数。可也没有明确指出对较旧参数的支持还将延续多长时间，因为很难预测将来发生的变化。从旧参数过渡到新参数这段时间内，您有足够的时间来规划如何将更改应用于应用程序。

在“新增内容”文档中首次讨论的产品也有一些新功能或功能部件。

如果您当前正在使用这些产品，则建议不要使用的某些功能或功能部件或者新的功能和功能部件将对您造成影响。对这些影响进行了概述，它们也是有关迁移的讨论的一部分。

下面是 DB2 的当前发行版与先前发行版之间的差别的列表。

系统目录信息:

将来版本的 DB2 中的 PK_COLNAMES 和 FK_COLNAMES:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

SYSCAT.REFERENCES 的列 PK_COLNAMES 和 FK_COLNAMES 不再可用。

症状:

引用时，由于列不再存在而返回错误。

解释:

这些列已过时，并且已被替换。

解决方案:

更改引用 SYSCAT.REFERENCES 的列 PK_COLNAMES 和 FK_COLNAMES 的工具或应用程序以取而使用 SYSCAT.KEYCOLUSE 视图。

将来版本的 DB2 中 COLNAMES 不再可用:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

SYSCAT.INDEXES 的列 COLNAMES 不再可用。

仅当列名小于或等于 30 个字节并且仅当索引中的列数小于或等于 16 时，此列才包含有效信息。如果任何列名超过 30 个字节，或者如果有多于 16 列，则将会返回空白或 NULL 值。

症状:

列不存在，并返回错误。

解释:

编码的工具或应用程序使用过时的 COLNAMES 列。

解决方案:

更改该工具或应用程序，以改用 SYSCAT.INDEXCOLUSE 视图。

应用程序编程:

不推荐使用 db2Load API 的 iCheckPending 参数:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

不推荐使用 db2load API 的 iCheckPending 参数。替换参数为 iSetIntegrityPending。

解释:

不推荐使用 db2Load API 的 iCheckPending 参数。它是 db2Load API 的输入参数，用于指定是否应使表处于检查暂挂状态。

注: 设置完整性暂挂状态将替换检查暂挂状态。它们是等价状态。

解决方案:

将 iSetIntegrityPending 参数与 db2Load API 配合使用。与此新参数配合使用的值有: SQLU_SI_PENDING_CASCADE_IMMEDIATE 或 SQLU_SI_PENDING_CASCADE_DEFERRED。

将不推荐使用用户定义的函数（UDF）和过程:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

不推荐使用下列 UDF: GET_DBM_CONFIG、SNAP_GET_CONTAINER、SNAP_GET_DB、SNAP_GET_DYN_SQL、SNAP_GET_STO_PATHS、SNAP_GET_TAB、SNAP_GET_TBSP、SNAP_GET_TBSP_PART、SNAPSHOT_AGENT、SNAPSHOT_APPL、SNAPSHOT_APPL_INFO、SNAPSHOT_BP、SNAPSHOT_CONTAINER、SNAPSHOT_DATABASE、SNAPSHOT_DBM、SNAPSHOT_DYN_SQL、SNAPSHOT_FCM、SNAPSHOT_FCMNODE、SNAPSHOT_LOCK、SNAPSHOT_LOCKWAIT、SNAPSHOT_QUIESCERS、SNAPSHOT_RANGES、SNAPSHOT_STATEMENT、SNAPSHOT_SUBSECT、SNAPSHOT_SWITCHES、SNAPSHOT_TABLE、SNAPSHOT_TBREORG、SNAPSHOT_TBS、SNAPSHOT_TBS_CFG 和 SQLCACHE_SNAPSHOT。

不推荐使用下列过程: GET_DB_CONFIG、SNAPSHOT_FILEW 和 SYSINSTALLROUTINES。

建议不要使用这些例程就意味着将不再继续开发这些例程。已经更新了这些例程的文档，以指示建议不要使用这些例程，但是，为了保持兼容，仍然要维护它们。在将来某个时候，将从目录和文档中除去这些 UDF 和例程。

解释:

基于 SQL 管理 API 标准的新例程将替换在实施标准之前创建的旧函数。

解决方案:

提供具有以 SNAP_GET_ 开头的类似名称的新等价函数。不同的参数和其他列可能与新函数关联。在应用程序内使用替换函数之前，应查看它们之间的差别。

有关不推荐使用的 UDF 和过程以及新的等价函数和视图的更多信息，请参阅"不推荐使用的 SQL 管理例程及其替换例程或视图"。

使用新函数、例程和视图；应考虑将使用旧函数和过程的应用程序更新为使用新函数和例程。为了保持兼容性，将继续支持旧函数和过程。但是，在产品的将来版本或发行版中将除去此支持。

已建议不要在外部例程库中使用缺省函数入口点:

受影响的操作系统:

仅 32 位 AIX 和 Windows 操作系统受影响。

更改:

在将来的某些版本或发行版中，将不再支持装入库名，而是假定使用缺省入口点。

在 AIX 和 Windows 操作系统环境内，已推荐不再支持外部例程中的缺省函数入口点。

解释:

当例程以可信（不受防护）方式运行时，如果仅指定库名并使用缺省入口点，则可能会发生实例故障。

解决方案:

从此以后，在创建存储过程和函数时，不依赖于数据库管理器来解决和装入缺省入口点指定的函数。相反，在装入例程库时指定完整的入口点和库名。对于新例程，指定 *!proc-id*（对于过程）或 *!func-id*（对于函数）值作为 EXTERNAL NAME 子句值的一部分。对于现有例程，为指定 EXTERNAL NAME 子句的例程定义提供一个显式入口点值。这可以使用 ALTER FUNCTION 语句来完成。

除去 1 类索引支持:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

除去了 1 类索引支持。

解释:

在版本 8 中引入了新类型的索引，称为 2 类索引。对于 1 类索引（该索引是在版本 8 之前创建的），键是在删除或更新表行时从叶子页中物理除去的。对于 2 类索引，当删除或更新行时，键标记为已删除，但在落实删除或更新前不会物理除去它们。当除去对 1 类索引的重新创建的支持后，将不必手工重建索引。1 类索引将继续正常工作。由索引的重新创建导致的所有操作将自动将 1 类索引转换为 2 类索引。在将来的版本中，将除去对 1 类索引的支持。

解决方案:

使用更新的 2 类索引。这可以通过手工转换较旧的索引来实现（通过在重组索引期间请求）。所有新索引都使用新的 2 类索引。

不推荐使用 DB2 JDBC 2 类驱动程序:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

在版本 8.2 中已不推荐使用 DB2 JDBC 2 类驱动程序，并且在版本 9.1 中继续不推荐使用该驱动程序。在将来的发行版中，将除去对该驱动程序的支持。

解决方案:

使用 IBM DB2 JDBC 和 SQLJ 驱动程序。

除去类型 3 JDBC 驱动程序支持:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

除去了类型 3 JDBC 驱动程序支持。

解释:

db2jd 将不随产品交付。

解决方案:

使用 IBM DB2 JDBC 和 SQLJ 驱动程序。

应用程序库已更改:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

进行了下列更改:

- 扩展了 db2app.dll。它包括其原始信息，以及 db2util.dll、db2abind.dll 和 db2cli.dll 库中的信息。
- 扩展了 db2api.dll。它包括其原始信息，以及 db2cli.dll 库中的信息。

解释:

将合并库信息。

解决方案:

仍可使用 db2util.dll、db2abind.dll 和 db2cli.dll 库的存根以实现向后兼容性。这些存根在产品的将来版本或发行版中会被除去。

***SQL*:**

已替换一些 SQL 管理例程:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

现有的某些管理例程已经被更新且更复杂的例程或视图所替代。

解释:

在版本 9 中，为了扩展对 SQL 管理例程的支持，需要替代现有的某些例程。

解决方案:

应该将使用版本 8 中的表函数的应用程序修改为使用新函数或管理视图。新的表函数与原始函数的基本名称相同，但是在它们后面添加了"_Vxx"来标识添加了这些函数的产品的版本。但是，管理视图将始终基于最新版本的表函数，因此，允许它们具有更高的应用程序可移植性。

有关新例程的更多信息，请参阅"不推荐使用 SQL 管理例程及其替换例程或视图"。

术语由"分区键"更改为"分布键":

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

术语"分区键"已更改为"分布键"。分布键是一列（或一组列），用来确定存储特定数据行的数据库分区。表分区键是一列或多列的有序集合，用来确定每个表行所属的数据分区。

解释:

引入表分区将要求重新定义"分区键"。

解决方案:

"分布键"用于文档中，以前使用的是"分区键"。

ALTER TABLE 语句中的 PARTITIONING KEY 子句发生了更改:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

建议不要在 ALTER TABLE 语句中使用 ADD PARTITIONING KEY 子句。此子句已被 ADD DISTRIBUTE BY HASH 子句取代。

建议不要在 ALTER TABLE 语句中使用 DROP PARTITIONING KEY 子句。此子句已被 DROP DISTRIBUTION 子句取代。

解释:

引入表分区将要求重新定义“分区键”，从而导致 ALTER TABLE 语句的语法更改。

解决方案:

为了保持向后兼容性，目前仍支持语法中使用旧的 PARTITIONING KEY 子句。但是，在产品的将来发行版中将不再支持该子句。因此，应计划将使用旧语法的应用程序转换为使用新语法。

***数据库安全性和调整*:**

不再支持扩充存储器:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

不再支持扩充存储器。将 DB2 产品迁移到 64 位环境中之后，就不需要扩充存储器了。

在迁移至 DB2 版本 9 之后，目录视图中的值将更改。例如，SYSCAT.BUFFERPOOLS 目录视图内的 ESTORE 列将始终为“N”。允许运行任何数据定义语言（DDL）来尝试更改此值，但这样做不起作用。

仍存在一些在版本 9 中不推荐使用的监视元素。四个监视元素为:

- *pool_data_to_estore*
- *pool_index_to_estore*
- *pool_data_from_estore*
- *pool_index_from_estore*

在 DB2 产品将来的发行版或版本中，将使用与扩充存储器相关的监视器元素，当请求 GET SNAPSHOT 命令时，将不再提供由这些元素的内容所生成的输出。

此外，在版本 9 中，用于扩充存储器的配置参数（**estore_seg_sz** 和 **num_estore_segs**）不再有效。

在将来的发行版或版本中，还将除去 SYSCAT.BUFFERPOOLS 目录中的“ESTORE”列。

解释:

扩充存储器充当主缓冲池的扩充后备缓冲区。这样可以提高内存性能，它利用了具有大量主存储器的计算机。对于具有 64 位环境的计算机，不再需要扩充存储器和其他类似的方法。

解决方案:

不应再使用扩充存储器。应计划不使用扩充存储器配置参数，也不使用扩充存储器监视器元素。

实用程序和工具:

不再支持桌面图标和文件夹生成实用程序（Linux）:

受影响的操作系统:

仅 Linux 操作系统受影响。

更改:

此发行版不再包括这样一组实用程序：它们用于创建 DB2 桌面文件夹和图标，以便为受支持的基于 Intel® 的 Linux 分发产品在 Gnome 和 KDE 桌面上启动常用的产品工具。

db2ilist 命令具有不推荐使用的选项（Linux 和 UNIX）:

受影响的操作系统:

仅 Linux 和支持的 UNIX 操作系统受影响。

更改:

db2ilist 命令具有下列不推荐使用的命令选项:

- *-w*（列示每个实例的位宽）
- *-a*（列示常规和 AFP™ 实例）
- *-p*（列示每个实例的路径）

解释:

过去，**db2ilist** 命令可用于列示系统上提供的所有实例。现在，**db2ilist** 命令仅列示与当前安装路径相关的实例，并且仅列示每个 UNIX 或 Linux 平台上的一种类型的实例。

解决方案:

仍可以使用 **db2ilist** 命令。不应使用不推荐对该命令使用的选项。

用于转换 DMS 表空间大小的 db2reg2large 实用程序不再可用:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

在 DB2 版本 9 中已不继续使用 db2reg2large 实用程序，该实用程序用于将常规 DMS 表空间转换为大型 DMS 表空间。

解决方案:

此实用程序已替换为 ALTER TABLESPACE SQL 语句中新的 CONVERT TO LARGE 选项。

不继续使用 **db2profc** 和 **db2profp** 实用程序:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

在前发行版中，已接受 db2profc 作为 db2sqljcustomize 的备用名称，并且已接受 db2profp 作为 db2sqljprint 的备用名称。不再接受这些备用名称。

解释:

DB2 JDBC 2 类驱动程序最初将名称 db2profc 用于 SQLJ 概要文件定制程序命令，并将名称 db2profp 用于 SQLJ 概要文件打印机命令。

解决方案:

对于 IBM DB2 JDBC 和 SQKJ 驱动程序，将 SQLJ 概要文件定制程序命令命名为 db2sqljcustomize，并将 SQLJ 概要文件打印机命令命名为 db2sqljprint。使用这些命令代替 db2profc 和 db2profp。

不推荐使用“设置数据库对象的许可权”（**db2secv82**）:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

不推荐使用“设置数据库对象的许可权”（**db2secv82** ）命令。

解释:

根据该命令的名称（db2sec**v82**）就知道，它仅用于产品的版本 8.2 。而在当前发行版和将来的发行版中将使用新名称。

解决方案:

使用“设置数据库对象的许可权”（**db2extsec**）命令来取代“设置数据库对象的许可权”（**db2secv82**）命令。应在应用程序和脚本中找到并更改对 **db2secv82** 命令的引用，制订计划来将这些引用替换为对 **db2extsec** 命令的引用。

db2look 工具的行为发生了变化:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

在使用“数据库分区功能”（DPF）的系统上，如果某些数据库分区处于不活动状态，则表空间数据定义语言（DDL）可能不完整。当在使用 DPF 的系统上请求 DDL 时，将显示一条警告消息，而不是显示存在于不活动数据库分区上的表空间的 DDL。

解释:

在多个数据库分区中使用自动调整大小和自动存储器，并因此而需要使用快照方法来收集数据时，将要求每个数据库分区都处于活动状态。

解决方案:

为了确保为所有表空间生成正确的 DDL，必须激活所有数据库分区。

忽略并不推荐使用 db2icrt、db2ilist 和 db2iupdt 命令的 WordWidth 参数（-w 选项）:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

使用 **db2icrt**、**db2ilist** 和 **db2iupdt** 命令的 WordWidth（*-w*）选项时已忽略它们并且将不推荐使用这些选项。此选项提供了实例宽度（按“位”计）。

解决方案:

如果继续指定此选项，不会有任何影响。该选项仅在 AIX 5L™、HP-UX、Linux 和 Solaris 操作系统上有效。

手工安装:

受影响的操作系统:

仅 Linux 和 UNIX 操作系统受影响。

更改:

不支持使用本机 Linux 或 UNIX 操作系统实用程序（例如，pkgadd、rpm、SMIT 或 swinstall）来手工安装、卸载或查询 DB2 产品。

解释:

为了更好地管理和控制安装过程，不再支持手工安装或卸载 DB2 产品。

解决方案:

使用具有新参数的 **db2_install** 命令来支持新函数。**db2_deinstall** 命令是基本安装映像的一部分。

将除去对“锁定对象名”的支持:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

“锁定对象名”是快照监视器样本输出的一部分，它不提供值，但包含输出的“锁定名称”部分中的冗余信息。在将来发行版中不推荐使用监视器元素“lock_object_name”。

解释:

快照监视器产生的输出报告生成锁定列表。“锁定名称”是该列表中的第一项。此信息来自监视器元素“lockname”。在报告的后面部分中，将显示“锁定对象名”。此信息来自监视器元素“lock_object_name”。此项中呈视的部分信息还可以从为监视器元素“lockname”给定的值中抽取出来。

解决方案:

在将来发行版中不推荐使用监视器元素“lock_object_name”。它提供的信息也将从快照监视器输出中除去。

应该计划不使用 GET SNAPSHOT FOR LOCKS ON <dbname> 命令来在任何新应用程序或修改过的应用程序中返回“锁定对象名”。

除去原始日志支持:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

除去了用于日志记录的原始设备支持。

解释:

由于专用存储子系统不断增多，并且完全支持自管理的 DMS 存储器，从而导致对详细存储管理的需求不断减少。

解决方案:

不要使用原始设备进行日志记录。可能需要将数据库配置参数设置 `newlogpath` 更改为磁盘设备而不是原始设备。请记住要停止数据库管理器，然后重新启动它以使配置参数的新设置生效。

对 db2batch 的更改:

受影响的操作系统:

支持的所有操作系统都受影响。

症状:

db2batch 命令现在仅以 CLI 方式运行。CLI 方式通常是使用 *-cli* 选项指定的。动态嵌入式 SQL 是缺省方式，但此方式已被更改，以便该命令仅以 CLI 方式运行。另外，对 db2batch.bnd 执行 REBIND 或 BIND 的脚本将失败，因为不再交付“bnd”文件。

另外，*-p* 选项也不可用。

解释:

不再支持 **db2batch** 命令中的并行选项。

解决方案:

只能继续使用 *-cli* 选项以实现向后兼容性，但这不起作用。可以通过在 `db2cli.ini` 文件中指定 *TxnIsolation* 配置关键字来更改缺省隔离级别。新的 *-iso* 选项用于指定隔离级别。

不能再在 **db2batch** 命令中使用 *-p* 选项。

将除去对 db2uiddl 工具的支持:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

将不再支持不支持延迟唯一语义的索引。在 DB2 版本 9 之后，将不再支持 **db2uiddl – 准备将唯一索引转换为 V5 语义**命令。

解释:

在 DB2 通用数据库（UDB）版本 5 中，已将唯一索引的语义更改为延迟唯一。为了支持这种更改，引入了 **db2uiddl** 工具来将唯一索引转换为新的语义。当迁移使用版本 5 之前的唯一索引语义的数据库时，并非将所有唯一索引都自动更改为版本 5 语义，因为转换唯一索引操作需要大量时间，您想要根据业务需求来管理转换。

解决方案:

在除去对 **db2uiddl** 的支持之前，制订一个计划以将在 DB2 UDB 版本 5 之前创建的所有唯一索引转换为新的延迟唯一索引语义。**db2uiddl** 工具将搜索系统目录以找到未使用延迟唯一语义的索引并为需要转换的索引编写 CREATE UNIQUE INDEX 语句。这些语句存储在一个文件中，在成功迁移至版本 9 之后，必须运行该文件。这将确保在不建议使用 **db2uiddl** 工具之前转换索引。

将除去对 db2undgp 工具的支持:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

现在，例程（函数、过程和方法）的 EXECUTE 特权由 SYSCAT.ROUTINEAUTH 系统目录视图控制。在 DB2 版本 9 之后，**db2undgp** 命令将不再可用。

解释:

在 DB2 通用数据库（UDB）版本 8 中，添加了系统目录视图 SYSCAT.ROUTINEAUTH 以控制例程（函数、过程和方法）的 EXECUTE 特权。在将数据库迁移至 DB2 UDB 版本 8 期间，为所有用户（PUBLIC）授予对所有现有函数、方法和外部存储过程的 EXECUTE 特权。这将导致访问 SQL 数据的外部存储过程的安全性暴露。**db2undgp** 命令用于防止没有特权的用户访问 SQL 对象。

解决方案:

在不推荐使用 **db2undgp** 工具之前，制订一个计划以从 PUBLIC 组调用 EXECUTE 特权。

将除去对类型 1 索引的重新创建的支持:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

在版本 8 中引入了新类型的索引，称为 2 类索引。对于 1 类索引（该索引是在版本 8 之前创建的），键是在删除或更新表行时从叶子页中物理除去的。对于 2 类索引，当删除或更新行时，键标记为已删除，但在落实删除或更新前不会物理除去它们。当除去对 1 类索引的重新创建的支持后，将不必手工重建索引。1 类索引将继续正常工作。由索引的重新创建导致的所有操作将自动将 1 类索引转换为 2 类索引。在将来的版本中，将除去对 1 类索引的支持。

解释:

2 类索引与 1 类索引相比较，具有以下优点:

- 可对长度大于 255 个字节的列创建 2 类索引。
- 使下一键锁定的使用减小到最小，这会改善并发性。

解决方案:

开发一个计划以在一定时间内将现有索引转换为 2 类索引。当最小化可用性停止时，“联机索引重组”功能可以帮助完成此操作。如果需要的话，增加索引表空间大小。考虑在大型表空间中创建新的索引，并将现有索引移至大型表空间。

注: 如果将版本 5 之前的索引转换为 2 类索引，则不需要运行 **db2uiddl** 工具。

连接和共存:

不再支持 CLI 关键字 CLISCHEMA:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

对于连接至 DB2 Linux 版、UNIX 版和 Windows 版 DB2 数据库服务器的 DB2 客户机，不推荐使用 CLISchema 关键字。

对于连接至 DB2 z/OS 版数据库服务器的 DB2 客户机，已删除 CLISchema 关键字。

解决方案:

DB2 客户机应不再使用 CLISchema 关键字。类似于 CLISchema 的一个关键字是 SysSchema。

不再包括数据仓库管理器:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

对于此发行版，DB2 仓库管理器标准版不可用。此发行版中未包括数据仓库中心和信息目录中心。

解决方案:

将这些产品和中心的开发和发布独立于基本 DB2 版本 9 产品。

不再支持 Text Extender:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

此发行版中不支持 DB2 Text Extender。

解决方案:

直接替换函数不可用。但是，有一些其他全文本搜索产品能够执行类似任务。例如，与 Text Extender 非常类似的 DB2 Net Search Extender；以及 WebSphere Information Integrator OmniFind™ 版，它提供用于查找大多数相关公司信息的企业搜索解决方案。不仅可以在关系数据库中找到这些信息，还可以在内部网、外部网、公司公共 Web 站点和大量内容存储库中进行搜索。

不再支持 Audio, Image, and Video (AIV) Extenders:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

此发行版中不再支持 Audio, Image, and Video (AIV) Extenders。

解决方案:

可以考虑使用DB2 用户定义的函数和第三方软件来实现自己的类似于 AIV Extenders 的扩展，以增强 DB2 功能。

DB2 管理工具的平台支持更改了:

受影响的操作系统:

仅支持的 Windows 和 Linux 操作系统受影响。

更改:

在前发行版中，DB2 管理工具（包括控制中心）在所有平台上都受支持。在版本 9 中，DB2 管理工具仅在 Windows x86、Windows x64（AMD64 或 EM64T）、Linux on x86 和 Linux on AMD64 或 EM64T 上受支持。

32 位实例支持发生了更改:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

为了满足市场需求，将优先考虑 64 位硬件和操作系统的 DB2 数据库服务器支持。受支持的 32 位平台数已减少。将继续支持 32 位 Windows 和 Linux 平台，因为构建或运行中小型业务应用程序时通常首选使用这些平台。

配置参数和注册表变量:

不推荐使用配置参数:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

不推荐使用下列配置参数:

- *estore_seg_sz*
- *num_estore_segs*
- *min_priv_mem*
- *priv_mem_thresh*; 使用 DB2MEMMAXFREE 注册表变量来代替它。
- *fcm_num_rqb*
- *fcm_num_anchors*
- *fcm_num_connect*

可以为这些配置参数中的每个参数设置值，但会忽略该值。(也就是说，值不起作用。)

不推荐使用注册表变量:

受影响的操作系统:

支持的所有操作系统都受影响。

更改:

不推荐使用下列注册表变量:

- DB2_FORCE_FCM_BP; 缺省值已从“否”更改为“是”。
- DB2_LGPAGE_BP; 使用 DB2_LARGE_PAGE_MEM 注册表变量来代替它。
- DB2LINUXAIO

不推荐使用下列注册表变量:

- DB2_SCATTERED_IO; 在 Linux 上，缺省值始终是从磁盘读取。

相关参考:

- 『Deprecated SQL administrative routines and their replacement routines or views』(*Administrative SQL Routines and Views*)

版本 9 与前发行版和已更改的行为不兼容

本节描述了 DB2 数据库系统的用户在编写新应用程序或修改现有应用程序时应记住的当前存在的不兼容性。这将有助于您当前进行应用程序开发和将来计划转为使用更新版本的 DB2。不兼容性通常涉及到产品功能和功能部件的缺省值发生更改；或者，它涉及到先前版本的 DB2 产品的不同需求或结果。例如，假定您在上一版本或发行版中使用了一个 SQL 语句，并期望它产生某种行为或结果。如果您在此版本中使用了同一 SQL 语句，但是接收到并不是您期望获得的另一种行为或结果，则说明上一版本（或发行版）与当前版本之间存在不兼容性。本节就说明了这些差异或“不兼容性”。

注: 尽管已尝试列示当前已知的所有产品不兼容性，但产品发行说明中可能记录了更新的不兼容性。

系统目录信息:

“设置”字段将更改:

Windows	UNIX

更改:

在特定目录中，“设置”字段的类型（和长度）发生了更改。

症状:

引用 ORDER BY 子句、WHERE SETTING IN... 子句或 WHERE SETTING= 子句的现有应用程序将失败。

解释:

下列目录中的设置文件已从 VARCHAR(255) 更改为 CLOB(32K):

- SYSCAT.TABOPTIONS
- SYSCAT.COLOPTIONS

由于 SQL 语句的大对象（LOB）所存在的限制，需要重写从这些目录中选择“设置”字段的应用程序。

解决方案:

重写从指定目录中选择“设置”字段的应用程序。

应用程序编程:

应用程序标识格式已更改:

Windows	UNIX

更改:

应用程序标识的格式已更改。

解释:

新格式按一种便于阅读的形式提供了端口号和 IP 地址，并且还接受更长的 IPv6 地址。

解决方案:

如果您具有用来对包含应用程序标识的输出进行解析的脚本，则需要修改解析条件以说明新格式。例如，可以解析 LIST APPLICATIONS 命令的输出。

DB2 嵌入式应用程序服务器已更新:

Windows	UNIX

更改:

DB2 嵌入式应用程序服务器允许您运行随 DB2 提供的 Web 应用程序，而无需购买应用程序服务器。在版本 8 中，DB2 嵌入式应用程序服务器也称为 *DB2 UDB 应用程序服务器*。

不再支持将 XML 元数据存储库（XMR）应用程序作为 DB2 嵌入式应用程序服务器的其中一个应用程序。

解决方案:

对于 XMR 版本 8 应用程序用户，必须卸载 XMR 并查找替换产品。WebSphere 提供了几个合适的替换产品。

现在支持 IBM Java 软件开发包 (SDK) 5.x:

Windows	UNIX

更改:

现在，下列操作系统平台可以支持 IBM Java 软件开发包（SDK）5.x: AIX 5、Linux on x86、Linux on AMD64/EM64T、Linux on zSeries®、Linux on POWER™、Windows x86 和 Windows x64。

解决方案:

IBM SDK 将自动安装在服务器上。如果安装了客户机工具，则在客户机上也将安装 IBM SDK。如果要对您自己的应用程序使用 JDBC 驱动程序，则需要确保安装了 IBM SDK。

应用程序和例程功能支持发生了更改:

Windows	UNIX

更改:

不再支持大多数 32 位数据库实例导致了对应用程序和例程的支持发生更改。

症状:

使用 DB2 版本 6 或版本 7 客户机实例的客户机应用程序不能连接至 DB2 版本 9 数据库服务器。

客户机应用程序环境中存在新的环境变量值。

在 DB2 通用数据库 版本 8 中创建的 32 位不受防护例程（存储过程和用户定义的函数）不能再在 AIX、HP、SUN、Linux on POWER、Linux for AMD64 and Intel EM64T 和 Linux on zSeries 环境中的 64 位 DB2 数据库服务器上运行。将这些例程迁移至 DB2 版本 9 需要在 64 位目标数据库服务器上重建它们。

为 DB2 通用数据库 版本 8（带有从修订包 1 到修订包 6 的任何修订包）的 32 位实例创建的 SQL 过程将不在 DB2 版本 9 的 64 位实例上运行。要成功将这些 SQL 过程迁移至 DB2 版本 9，必须使用 64 位目标数据库服务器删除并重新创建 SQL 过程。为带有任何修订包的 DB2 通用数据库版本 7 或版本 8 的 32 位实例创建的 SQL 过程将继续在 DB2 版本 9 的 32 位受支持实例上运行。

解释:

不再支持大多数 32 位数据库实例导致了对应用程序和例程的支持发生更改。

解决方案:

您需要考虑在产品更改后是继续使用 32 位数据库实例，还是应移至 64 位数据库实例。

例如，随 64 位 DB2 数据库服务器仅提供了 64 位 JVM。仅对 Linux x86 和 Windows on x86 操作系统提供了 32 位 JVM。最终，Java 外部例程需要 32 位 JVM 用于 32 位 DB2 数据库服务器，并且需要 64 位 JVM 用于 64 位 DB2 数据库服务器。

交付的新函数和过程:

Windows	UNIX

更改:

随产品交付的例程集中添加了新函数、现有函数的函数特征符或过程。

症状:

如果用户定义的函数或用户定义的过程的名称和特征符与交付的新函数或过程的名称和特征符相同，则动态 SQL 语句中对该函数或过程的未限定引用现在将执行交付的函数或过程，而不是用户定义的函数或过程。请注意，这并不会影响程序包中的静态 SQL 或将继续执行用户定义的函数或过程的 SQL 对象（例如，视图、触发器或 SQL 函数），直到显式绑定程序包或删除和创建 SQL 对象为止。

解释:

缺省 SQL 路径中的模式名前面包含模式 SYSIBM、SYSFUN、SYSPROC 和 SYSIBMADM，它是 USER 专用寄存器的值。使用 SET PATH 语句或 FUNCPATH 绑定选项显式设置 SQL 路径时，通常这些系统模式也会包括在该 SQL 路径中。执行函数解析和过程解析时，这些模式中交付的函数和过程将视为用户定义的函数和用户定义的过程。

在版本 9 中，已将下列函数和过程添加至交付的函数和过程集:

CHARACTER_LENGTH

OCTET_LENGTH

POSITION

SECLABEL

SECLABEL_BY_NAME

SECLABEL_TO_CHAR

STRIP

SUBSTRING

TRIM

XMLCOMMENT

XMLDOCUMENT

XMLQUERY

XMLTEXT

XMLVALIDATE

XMLXSROBJECTID

在版本 9 中，已添加新的管理函数和过程。由于对这些函数和过程使用的命名约定使得用户定义的函数或用户定义的过程不可能有相同的名称，所以此处未列示它们。有关这些函数和过程的列表，请参阅 *Administrative SQL Routines and Views*。

解决方案:

重命名用户定义的函数或用户定义的过程；或者使名称成为标准名称以调用它。否则，将使用交付的函数或过程。或者，可以将用户定义的函数或用户定义的过程所在的模式放置在 SQL 路径中交付的函数或过程所在的模式之前。但是，这样将增加解析所有交付的函数和过程所用的时间，因为会先考虑系统模式前的模式。

对 LIST APPLICATIONS 输出的更改:

Windows	UNIX

更改:

有两个新的代理程序将作为 LIST APPLICATIONS 命令的一部分。

解释:

有两个新的代理程序（db2stmm 和 db2taskd）需要一直与数据库相连。因此，有两个新的代理程序将作为 LIST APPLICATIONS 命令的一部分。如果您使用了任何脚本来监视 LIST APPLICATIONS 命令的输出，则需要根据这两个新的代理程序来修改这些脚本。

解决方案:

修改用来监视 LIST APPLICATIONS 命令的输出的任何脚本，以说明存在两个新的代理程序。

DMS 表空间的缺省大小:

Windows	UNIX

更改:

DMS 表空间的新缺省大小是“大型”。

症状:

如果具有用来创建 DMS 表空间并且不显式指定大小（无论是常规还是大型）的脚本，则使用的存储器大小可能会增多。旧的缺省值是“常规”，新的缺省值是“大型”。

解释:

DMS 表空间的缺省大小是“大型”，它比“常规”表空间所占用的空间要多。

解决方案:

对于用来创建表空间并且过去只接受缺省值的那些脚本，如果您希望的话，应该考虑通过添加“常规”表空间大小的显式请求来修改这些脚本。

SQL:

SQL 过程不再能使用游标分块:

Windows	UNIX

更改:

无论对 BLOCKING 绑定选项指定何值，也不再能对 SQL 过程使用游标分块。始终是一次接收一行数据。

解释:

对于 FETCH 语句和隐式包含的 FOR 循环中的 FETCH 语句，有一个新的限制适用。

解决方案:

查看使用游标分块的这些应用程序。可能需要根据行为上的这个更改修改这些应用程序。

锁列表需要其他空间:

Windows	UNIX

更改:

锁列表中的每个锁所需的空间已更改，这样给定大小的锁列表不再表示它过去所表示的那么多锁。

解释:

锁定大小已按如下所示更改:

- 在 32 位平台上，对于具有现有锁定的对象，记录该对象的每个锁需要 48 字节的锁列表。锁需求为 40 字节。
- 在 64 位 HP-UX/PA-RISC 系统上，对于具有现有锁定的对象，记录该对象的每个锁需要 80 字节的锁列表。锁需求为 64 字节。

解决方案:

可能需要修改锁列表大小。

新函数 SYSIBM.LOCATE 替换 SYSFUN.LOCATE:

Windows	UNIX

更改:

版本 9.1 中交付了一个新函数 SYSIBM.LOCATE，它扩展 SYSFUN.LOCATE 中提供的功能。

症状:

如果在不限定模式名的情况下，在版本 9.1 中编译的应用程序中使用 LOCATE 函数，则将调用新的 SYSIBM.LOCATE。在某些情况下，SYSIBM.LOCATE 返回的结果可能与 SYSFUN.LOCATE 函数的结果不同。

解释:

SYSIBM.LOCATE 通过将字符语义添加至 LOCATE 函数并接受图形字符串自变量，从而扩展了 SYSFUN.LOCATE 的功能。尽管 SYSFUN.LOCATE 所支持的现有语法仍有效，并且在使用 OCTETS 或不使用 CODEUNITS 规范时使用 SYSIBM.LOCATE 都具有相同的语义，但在较少情况下还是会产生不同结果。

对图形字符串数据类型执行搜索时就会出现这样一种情况。在版本 9.1 之前的发行版中，在 Unicode 数据库中，将在调用函数之前把图形字符串转换为字符串。在未指定 CODEUNITS 规范的情况下，SYSFUN.LOCATE 的搜索成功位置以字节为单位计数；而 SYSIBM.LOCATE 以两个字节为单位计数。

例如，下列语句返回值 2:

```
VALUES SYSIBM.LOCATE(GX'0040', GX'D8000040')
```

而下列语句返回值 4:

```
VALUES SYSFUN.LOCATE(GX'0040', GX'D8000040')
```

如果 2 字节的图形字符搜索字符串作为跨越源字符串中的两个“实际字符”的字节模式出现，则这两个函数之间也会有所不同。例如，使用 SYSFUN.LOCATE 在字符串 GX'11223344 中搜索 GX'2233 成功，但使用 SYSIBM.LOCATE 返回 0（未找到）。这是因为 SYSFUN.LOCATE 执行基于字节的搜索，而 SYSIBM.LOCATE 执行基于字符的搜索。源字符串中的字符是“1122”和“3344”。不存在字符“2233”，它只是出现的一种字节模式；但跨越两个字符。

另一个差别是 SYSIBM.LOCATE 会进行一些字符验证（而 SYSFUN.LOCATE 不会）并在搜索字符串中存在无效字符时给出不同的结果。

解决方案:

如果结果中由非法字符产生的差别可接受，则用户可以使用 OCTETS 作为代码单位规范来获得与 SYSFUN.LOCATE 相同的行为。如果需要 SYSFUN.LOCATE 函数的准确行为，则应用程序可以使用用模式名限定的函数名。建议您使应用程序适应新的 SYSIBM.LOCATE，因为它提供更多功能。

根据指定单位进行 SQL 函数处理:

Windows	UNIX

更改:

并非所有对字符串进行运算的 SQL 函数都被限制为处理"字节数"。

解释:

CHARACTER_LENGTH、LENGTH、LOCATE、POSITION 和 SUBSTRING 函数包括一个参数，它允许您指定一组预定义的字符串单元。这表示这些函数可以使用指定的单位而不是字节数或双字节数来处理字符串。

例如，与使用"字节数"的 SYSFUN 版本的 LENGTH 和 LOCATE 相比，SYSIBM.LENGTH 和 SYSIBM.LOCATE 函数处理"字符数"形式的输入。这将导致每个函数在处理非法字符时禁用不同的行为。

解决方案:

如果正在处理的数据中有非法字符，则在选择使用的函数时要特别小心。根据使用的函数，可以期望不同的结果。

创建索引时的新扫描缺省值:

Windows	UNIX

更改:

当创建新的主键、唯一键或索引（扩展索引除外）时，ALLOW REVERSE SCANS 是缺省值。因此，由于优化器可能在某些 SQL 语句中可以使用逆向索引扫描，所以访问方案可能会更改，查询的执行时间也可能会延长。但是，使用扩展索引类型时情况有所不同。在先前发行版中，使用的缺省值为 DISALLOW REVERSE SCANS。

解释:

新缺省值允许优化器同时考虑通过索引进行正向扫描和逆向扫描。

注: 如果对同一个表创建两个索引，一个索引指定升序顺序（ASC），另一个索引指定降序顺序（DESC），且在 CREATE INDEX 语句中不指定 DISALLOW REVERSE SCANS 选项，则这两个索引都将缺省为 ALLOW REVERSE SCANS。因此，将不会创建后一个索引，并且会发出一条警告消息指出索引发生重复。

解决方案:

在先前的版本中，您可能已经创建了一个正向扫描索引和一个逆向扫描索引来加快应用程序的速度。遗憾的是，这样做需要维护两个索引。既然缺省情况下将启用逆向扫描，就可以将两个索引替换为对逆向扫描启用的单个索引。

如果您不想对要创建的索引进行逆向扫描，则必须显式请求使用 DISALLOW REVERSE SCANS 选项来创建索引。

<u>创建新数据库时，缺省情况下将启用的新功能部件</u>:

Windows	UNIX

更改:

当创建新数据库时，缺省情况下将启用自调整内存、配置顾问程序和自动 RUNSTATS。

解释:

在创建新数据库之后，您可能会看到因这些自主功能部件的新缺省值发生更改而导致的不同查询方案或工作负载行为。依赖于先前的 DB2 缺省行为和数据库配置值的现有应用程序或脚本可能会发现更改，因为某些配置值已经被更改。

解决方案:

如果想在缺省情况下不启用这些功能部件，则可以通过对相应的功能部件执行操作来禁用它们:

- 配置顾问程序：在创建数据库之前，使用 **db2set** 将注册表变量 DB2_ENABLED_AUTOCONFIG_DEFAULT 设置为“NO”。
- 自调整内存：在创建数据库之后，通过将数据库配置参数 *self_tuning_mem* 设置为“OFF”来更新该参数。
- 自动 RUNSTATS：在创建数据库之后，通过将数据库配置参数 *auto_runstats* 设置为“OFF”来更新该参数。

<u>不允许在一个单元中对同一缓冲池进行多次更改</u>:

Windows	UNIX

更改:

不再允许同一个工作单元中具有多个 ALTER BUFFERPOOL 语句。

解释:

在版本 9 中，增加了自调整内存管理器，从而增加了可以对缓冲池的各种特征执行的操作的复杂性。为了降低这种复杂性，就不再允许在单个工作单元中多次改变同一缓冲池的各种特征。

解决方案:

不允许尝试在同一个工作单元中使用多个 ALTER BUFFERPOOL 语句。

<u>数据库安全性和调整</u>:

SET SESSION AUTHORIZATION 需要 SETSESSIONUSER 特权:

Windows	UNIX

更改:

在 DB2 版本 9 中，使用 SET SESSION AUTHORIZATION 语句将会话授权标识更改为一个新值要求 SQL 语句的授权标识具有 SETSESSIONUSER 特权。此特权可由安全管理员（SECADM）使用新的 GRANT SETSESSIONUSER 语句授予。

解释:

在 DB2 UDB 版本 8 中，具有 DBADM 或 SYSADM 权限的用户可以通过使用 SET SESSION AUTHORIZATION 语句对同一连接采用不同授权标识。在 DB2 版本 9 中，需要新的 SETSESSIONUSER 特权（只能由安全管理员（SECADM）授予）来执行此任务。

解决方案:

为了实现向后兼容性并避免现有用户特权丢失，在迁移至 DB2 版本 9 时，将自动授予显式拥有 DBADM 权限（如 SYSCAT.DBAUTH 目录视图中所记录的那样）的任何授权标识 SETSESSIONUSER 特权。除非对迁移至 DB2 版本 9 之后需要 DBADM 权限的用户显式授予 SETSESSIONUSER 特权，否则这些用户无法更改会话授权标识。

涉及管理类的 TSM 过滤更改:

Windows	UNIX

更改:

在 DB2 版本 9 之前，如果指定了管理类，则复原和日志检索可以根据它搜索对象。因为管理类可以更改，所以根据管理类进行过滤可能产生不正确的结果。因此，不再将管理类用做过滤基础。

解释:

管理类是一个 Tivoli Storage Manager（TSM）概念，它根据定义的存储策略帮助管理对象。将备份映像、装入副本映像或日志文件写入 TSM 时，一个特定管理类将与该对象关联。在写入日志文件或存储备份映像之后，管理类可能通过 TSM 更改。

解决方案:

不再将管理类用作过滤基础。

使表脱离设置完整性暂挂状态所需特权和权限的更改:

Windows	UNIX

更改:

SET INTEGRITY 和 REFRESH TABLE 语句需要特定权限和特权来处理受这些语句影响的表。从版本 8 到版本 9，授权标识所拥有的权限和特权列表已更改。

解决方案:

使表脱离设置完整性暂挂状态并执行需要的完整性处理要求特定权限和特权。语句授权标识拥有的权限和特权必须至少包括下列其中一项权限或特权:

- 对以下表的 CONTROL 特权:
  - 执行完整性处理的表，如果为一个或多个这种表提供了异常表，则还需要对异常表的 INSERT 特权
  - 通过语句隐式设置为“设置完整性暂挂”状态的所有派生外键表、派生立即具体化查询表和派生立即登台表
- LOAD 权限（具有条件）。必需先满足下列所有条件，才能将 LOAD 权限视为提供有效的特权:
  - 必需的完整性处理不涉及下列操作:
    - 刷新具体化查询表
    - 传播至登台表
    - 更新生成列或标识列
  - 如果为一个或多个表提供了异常表，则将在完整性处理期间授予对执行了完整性处理的表和相关联的异常表的必需访问权。即:
    - 对执行了完整性处理的每个表的 SELECT 和 DELETE 特权，以及
    - 对异常表的 INSERT 特权
- SYSADM 或 DBADM 权限

可能会导致错误的索引更改:

Windows	UNIX

更改:

一个索引中的最大列数已从 16 列增大到 64 列；因此，索引键的最大大小取决于索引页大小。如果发生排序堆溢出，则要求系统临时表空间具有足够大的页大小供排序使用。

解释:

在先前发行版中，使用 4 KB 页的缺省临时系统表空间可能就已经足够了。但是，在当前发行版中，键大小、记录标识与页标题大小之和可能超过了 4 KB 页，因此会产生错误消息 SQL1584N。

解决方案:

应该对可能会遇到此错误消息的应用程序进行更新，以便检测它并按照产生的消息而执行相应的操作。

不再支持扩充存储器:

Windows	UNIX

更改:

不再支持扩充存储器。将 DB2 产品迁移到 64 位环境中之后，就不需要扩充存储器了。

解释:

扩充存储器充当主缓冲池的扩充后备缓冲区。这样可以提高内存性能，它利用了具有大量主存储器的计算机。对于具有 64 位环境的计算机，不再需要扩充存储器和其他类似的方法。

解决方案:

不应再使用扩充存储器。如果您要使用 Windows 并且想要使用更多内存，则应考虑移至 64 位操作系统。但是，如果必须继续驻留在 Windows 32 位操作系统上，则可以使用“地址窗口扩展”（AWE）来解决 32 位空间局限性。AWE 受注册表变量 DB2_AWE 控制。

缺省情况下将数据库作为自动存储器来创建:

Windows	UNIX

更改:

在版本 9 中已经更改了 CREATE DATABASE 命令和 sqlecrea() API。现在，它们在缺省情况下将创建启用了自动存储器的数据库。您必须显式指定非自动存储器以使用旧的行为。

症状:

SYSCATSPACE、TEMPSPACE1 和 USERSPACE1 都将作为自动存储器表空间来创建。这就意味着数据库管理器将管理这些表空间的存储器。对自动存储器表空间执行容器操作是无效的。TableID（FID）也将被更改，尽管您可能对这些值不感兴趣。重定向复原操作也会对自动存储器具有不同的影响（即，重新定义存储路径而不是单个表空间容器）。

解释:

自动存储器将管理自动存储器表空间的容器，因此，不能对这些表空间执行某些操作（例如，容器操作和重定向复原）。注意，此更改并不会影响作为 SMS 或 DMS 来显式创建的表空间。从先前发行版迁移而来的数据库也不会受影响。这可能会影响依赖于缺省表空间的特征的那些脚本。

由于表空间类型发生了更改，因此，磁盘需求将增大。缺省情况下，非临时自动存储器表空间一次将增大 32 MB，因此，小型数据库可能会拥有更多磁盘空间。当数据库增大时，就将使用此空间。同样，空表也将使用更多空间。空表和索引将占用 512 KB。可以通过更改表空间的扩展数据块大小来减小此空间，可以显式更改扩展数据块大小，也可以通过修改数据库配置中的缺省扩展数据块大小（DFT_EXTENT_SZ）来更改扩展数据块大小。对于小型数据库，建议将扩展数据块大小设置为 4。仅当创建了表空间时才能选择扩展数据块大小。

解决方案:

必要时，可以通过调用附带了 AUTOMATIC STORAGE NO 子句的 CREATE DATABASE 或者调用附带了 SQL_AUTOMATIC_STORAGE_NO 的 sqlecrea() 来创建非自动存储器数据库。还可以更新应用程序以正确使用新的表空间类型；例如，重新定义复原时的存储路径，而不是发出 SET CONTAINERS 并作为重定向复原的一部分。

实用程序和工具:

不再支持自动装入器实用程序（db2atld）:

Windows	UNIX

更改:

不再支持自动类装入器实用程序（db2atld）。

解释:

现在，在分区数据库环境中，建议使用 LOAD 实用程序来分配和装入数据。

解决方案:

在分区数据库环境中，使用 LOAD 实用程序来分配和装入数据。

从游标装入:

Windows	UNIX

更改:

使用当前发行版中的 CURSOR 文件类型和分区数据库配置装入选项 PARTITION_ONLY 执行装入操作时，不能使用前发行版中的分布式数据文件。

解释:

分布式数据文件与新的 DB2 服务器不兼容。反之亦然；即使用 CURSOR 文件类型和分区数据库配置装入选项 PARTITION_ONLY 执行装入操作时，不能使用当前发行版中的分布式数据文件。

解决方案:

使用 CURSOR 文件类型和分区数据库配置装入选项 PARTITION_ONLY 对版本 9 DB2 服务器执行装入操作时，必须使用通过 DB2 版本 9 创建的一组分布式数据文件。

提供方装入 API（sqluvtld）不再有效:

Windows	UNIX

更改:

提供方装入 API（sqluvtld）不再可供使用。

解释:

现在，建议使用 Load 实用程序来分布和装入数据。

解决方案:

Load 实用程序是唯一受支持的批量装入程序。可以使用 db2Load API 来运行 Load 实用程序。

对 db2batch 的更改:

Windows	UNIX

更改:

动态嵌入式 SQL 是缺省方式，但此方式已被更改，以便该命令仅以 CLI 方式运行。通过提供附加信息（例如，时间戳记和清除器消息）改进了 **db2batch** 命令提供的输出。该输出也是采用新格式。

解释:

不再支持 **db2batch** 命令中的并行选项（*-p*）。

解决方案:

只能继续使用 *-cli* 选项以实现向后兼容性，但这不起作用。可以通过在 `db2cli.ini` 文件中指定 *TxnIsolation* 配置关键字来更改缺省隔离级别。新的 *-iso* 选项用于指定隔离级别。

不能再在 **db2batch** 命令中使用 *-p* 选项。

更改了分布式数据文件的 LOAD 命令集:

Windows	UNIX

更改:

在版本 9 中，不能将在先前版本中使用 LOAD 命令创建的一组分布式数据文件（这些文件使用 CURSOR 文件类型，并且指定了分区数据库配置装入选项 PARTITION_ONLY）用作 LOAD 命令的输入。即，先前创建的分布式数据文件与使用 CURSOR 文件类型和分区数据库配置装入选项 LOAD_ONLY 的 LOAD 命令不兼容。

解释:

分布式数据文件的格式在版本 9 中已更改。

解决方案:

当创建一组分布式数据文件时，应将数据分区，并使用同一版本的 DB2 产品来装入数据。

对 db2ckmig 工具的更改:

Windows	UNIX

更改:

如果 **db2ckmig** 工具返回的消息的 SQLCODE 存在，则 **db2ckmig** 日志文件现在同时包括 SQLCODE 和 SQL 消息文本。

解释:

在先前发行版中，**db2ckmig** 工具使用它自己的消息文件中的消息文本来报告错误。但是，在某些情况下，现有 SQLCODE 也描述错误。在 **db2ckmig** 日志文件中包含 SQLCODE 表示您可以参阅消息文档以获取对问题的更详细解释和可能的用户响应。

解决方案:

在 **db2ckmig** 日志文件的准确消息文本基础上构建的任何工具可能需要更改以解析 SQLCODE。

REORGCHK 命令输出更改:

Windows	UNIX

更改:

对于版本 9，作为 REORGCHK 命令的一部分生成的输出已更改。

SCHEMA 列和 NAME 列并置成一列（SCHEMA.NAME）。此外，每个表和索引的 SCHEMA.NAME 都分为两行，一行表示表的实际标准名称，一行表示该表的每个索引的标准名称。其余列的实际数据在每个索引名后面。

解决方案:

您可能必须考虑对 REORGCHK 命令的输出所进行的更改。

对迁移支持工具和命令的更改:

Windows	UNIX

更改:

所提供的数据库工具、实用程序和命令支持从版本 8 进行迁移，但不支持从版本 7 进行迁移。

解释:

从版本 7 迁移到版本 8，再从版本 8 迁移到版本 9 这一过程涉及到的更改很多并且非常复杂，因此，使用一组迁移工具和命令很难完成从版本 7 直接迁移到版本 9。

解决方案:

使用迁移信息来规划如何迁移到当前版本和发行版。这可能涉及到在尝试迁移到当前版本和发行版之前先迁移到版本 8。

备份映像的新命名约定:

Windows	UNIX

更改:

已更改存储在 Windows 操作系统上的备份映像的命名约定，以便与用于所有其他操作系统的命名约定匹配。

解释:

现在，在磁盘上创建的备份映像文件名由几个元素并置组成，这些元素由句点分隔:

```
DB_alias.Type.Inst_name.NODEnnnn.CATNnnnn.timestamp.Seq_num
```

解决方案:

将此新命名约定用于备份映像。

注: 在 DB2 V9.1 数据库系统上，仍可以复原使用先前命名结构的先前版本的产品中的备份映像。

对 **db2look** 输出的更改:

Linux	UNIX

更改:

在 **db2look** 命令生成的输出中，现在对标识整理顺序显示的值是 IDENTITY。

解释:

在前发行版中，对 **db2look** 命令和 GET DATABASE CONFIGURATION 命令所生成的输出中的标识整理顺序显示的值是 BINARY。整理顺序本身未更改。

对数据移动实用程序的更改:

Linux	UNIX

更改:

已对 Load、Import 和 Export 实用程序进行了下列更改:

- 使用 IXF 文件格式重新创建表时，如果在使用 CREATE 选项执行导入过程中无法重新创建某个功能，则在导出过程中将接收到一个警告，并在导入过程中接收到错误。在某些情况下，可以通过指定文件类型修饰符 FORCECREATE 强制从 IXF 文件创建表。这种新行为只对使用 DB2 版本 9.1 导出的文件产生影响。
- 现在，导出的 LOB 文件的扩展名是 `.lob`。例如，filename.001.lob 和 filename.002.lob。LOB 文件的缺省名称是输入数据文件名。例如，<datafile>.001.lob 和 <datafile>.002.lob。如果输入数据文件是在 DB2 UDB 版本 8 中生成的，则 DB2 版本 9.1 Import 实用程序可以正确地读取该文件。

- 移动 LOB 数据时，缺省路径和 Load、Import 和 Export 实用程序搜索这些路径的顺序已更改。
- 导出和导入 LOB 数据时，如果在 EXPORT 命令中指定 LOBS TO 或 LOBFILE 选项，或者在 IMPORT 命令中指定 LOBS FROM 选项，则会自动指定 LOBSINFILE 关键字。在 DB2 UDB 版本 8 中，如果未指定 LOBSINFILE 文件类型修饰符，则将忽略 LOBS TO、LOBS FROM 和 LOBFILE 选项。

对 **db2mtrk** 命令的更改:

Linux	UNIX

更改:

现在，Windows 平台上支持显示数据库级别内存的 -d 选项。显示实例级别内存的 -i 选项不再显示数据库级别内存。

解释:

由于 -d 选项现在在 Windows 平台上可用，所以应使用它来显示数据库级别内存。使用 -i 选项时，将仅显示实例级别内存。

解决方案:

在 Windows 平台上，使用 **db2mtrk** 命令的 -d 选项来查看数据库级别内存。

对用于自动维护的诊断消息的位置更改:

Linux	UNIX

更改:

已对与自动维护相关的诊断级别和消息位置进行了下列更改:

- 每次评估自动维护运行状况指示器时，就将诊断记录写入 db2diag.log 文件中。如果这些评估的结果导致执行维护操作，则诊断记录会同时写入 db2diag.log 文件和通知日志中。
- 与自动维护关联的诊断记录分类为“信息”记录。
- 仅当实例的诊断级别（diaglevel）或通知级别（notifylevel）设置为值 4 时，才写入这些诊断记录。

解释:

在 DB2 通用数据库 版本 8 中，每次评估自动维护运行状况指示器时，就会将诊断记录写入 db2diag.log 文件中。每次这些评估的结果导致执行维护操作时，就会将另一个条目写入 db2diag.log 文件中。这些诊断记录分类为“事件”记录，将实例的诊断级别（如 diaglevel 数据库管理器配置参数中所指定的那样）设置为值 3 或 4 时才会显示这些记录。

解决方案:

要确保诊断记录（“事件”记录）出现在 db2diag.log 文件和通知日志中，将实例的诊断级别（diaglevel）或通知级别（notifylevel）设置为 4。

表空间时间点前滚操作的限制:

Linux	UNIX

更改:

对于 DB2 版本 9 客户机，必须在某个时间点执行所有表空间前滚恢复操作。

解决方案:

确保所有客户机已迁移至 DB2 版本 9，并确保在初始化前滚操作时指定了一个时间点。

写入表事件监视器更改:

Linux	UNIX

更改:

在分区数据库环境中，写入表事件监视器仅在包含事件监视器表的表空间所在的数据库分区上处于活动状态。当活动事件监视器的目标表空间不存在于特定数据库分区上时，事件监视器将在该数据库分区上取消激活，并且会将一个错误写入 db2diag.log 文件中。

解释:

在较早版本的 DB2 中，事件监视器将在这些数据库分区上处于活动状态并且作为活动事件监视器进程出现，但不写入任何数据。

连接和共存:

已增加日志、表空间和内存需求:

Linux	UNIX

更改:

已增加记录标识（RID）大小来支持大型表空间。还增加了日志文件的增长速率和日志记录的大小。现在，每个 RID 在单分区环境中需要 8 字节的内存，而在分区数据库环境中需要 16 字节的内存。

解释:

随着与较大记录标识（RID）有关的相关更改，对日志、表空间和内存的需求也在不断增加。较大记录标识（RID）允许每个表对象有更多数据页，每个页有更多条记录。页数和记录数的增加还会更改日志文件和系统临时表空间使用的内存和空间大小。

解决方案:

如果结果集中的行大小接近具有最大页大小的现有系统临时表空间的最大行长度限制，则可能需要创建具有较大页大小的系统临时表空间。另一种方法是减短查询检索的信息的长度或分割查询。

数据库需要其他空间:

Linux	UNIX

更改:

与前发行版中的对象相比，适应 DB2 产品中的更改需要您为这些数据库对象分配更多空间。

解释:

此版本的 DB2 产品中的更改表示日志、表空间、索引、系统目录表和用户表数据需要其他空间。

解决方案:

在创建数据库对象之前，查看对这些对象所作的更改，以便了解增加的空间需求。

Linux 和 UNIX 上的 DB2 安装映像的程序包格式更改:

Linux	UNIX

更改:

Linux 和 UNIX 上的 DB2 安装映像不再使用操作系统格式。

不能再使用 Linux 和 UNIX 操作系统实用程序（例如，pkgadd、rpm、SMIT 或 swinstall）。

解释:

为了使您能够在同一系统上安装多个 DB2 副本，将 Linux 和 UNIX 上的所有 DB2 安装映像压缩成 tar.gz 格式。

解决方案:

您应使用 DB2 安装程序来确保正确部署并设置了 DB2 产品。如果您有过去用于使用操作系统命令来安装 DB2 产品的脚本，则必须修改这些脚本以调用 DB2 安装程序（**db2setup** 或 **db2_install**）。

只能使用 **db2ls** 命令来查询 DB2 产品的安装。如果使用了包含操作系统命令的脚本来查询 DB2 安装程序包，则必须修改这些脚本以使用 **db2ls**。

不再支持 NetBIOS 和 SNA:

Windows	UNIX

更改:

不再支持将 NetBIOS 和 SNA 作为数据库系统之间的通信方法。

解释:

不再支持 NetBIOS 和 SNA。

解决方案:

不要计划使用 NetBIOS 或 SNA 作为将来数据库客户机与服务器之间的通信方法。从 DB2COMM 注册表变量中除去 NetBIOS 和 SNA 关键字，以防止在启动实例时生成错误。使用 CATALOG NETBIOS NODE、CATALOG APPC NODE 或 CATALOG APPN NODE 命令时也会返回错误。

安装期间不再支持的 DB2 产品:

Windows	UNIX

更改:

不再支持下列产品作为安装选件或必备组件:

- DB2 数据仓库中心
- DB2 数据仓库管理器
- DB2 信息目录中心
- DB2 Data Links Manager
- DB2 Datajoiner

解释:

解决方案:

如果系统上安装了上述任何产品，则在将 DB2 数据库系统迁移至版本 9 之前必须卸载它们。如果安装了上述任何产品，则实例迁移将失败。

此外，在卸载 DB2 产品之后，这些产品创建的任何数据库对象（例如，用户定义的类型、用户定义的函数和存储过程）仍将保留在数据库中。在迁移之前应从数据库中除去这些对象，因为它们可能导致迁移失败。

不再支持 Data Links Manager:

Windows	UNIX

更改:

不再支持 DB2 Data Links Manager。因此，也不支持 Data Links 服务器的下列组件:

- Data Links 文件管理器（DLFM）
- 用来控制 Data Links 文件系统（DLFS）的 Data Links 文件系统过滤器（DLFF）
- DB2 日志记录管理器

解释:

不再支持 DB2 Data Links Manager。

解决方案:

不要创建数据类型为 DATALINK 或者引用 DATALINK 列的任何新数据库对象。

控制中心中不再支持 **VM/VSE** 对象:

Windows	UNIX

更改:

在 DB2 控制中心中，您无法再连接至 VM/VSE 数据库或与该数据库断开连接。可以仅显示已编目的 VM 和 VSE 数据库。添加实例时，VM 和 VSE 操作系统不再可供选择。

解决方案:

虽然可以显示已编目的 VM 和 VSE 数据库，但必须独立于 DB2 控制中心连接至它们。

使用的最大连接数已更改:

Windows	UNIX

更改:

有两个新的代理程序需要一直与数据库相连。

解释:

有两个新的代理程序（db2stmm 和 db2taskd）需要一直与数据库相连。这些连接需求意味着，在紧密配置的环境中，将一直使用由数据库管理器配置参数 *max_connections* 标识的两个连接。因此，您可能会用完可用连接。

解决方案:

如果是在一个紧密配置的环境中工作，则应考虑将数据库管理器配置参数 *max_connections* 的值加上 2。

配置参数和注册表变量:

配置参数缺省值更改:

Windows	UNIX

更改:

在 DB2 数据库版本 8.2 和版本 9 之间，下列配置参数的缺省值已更改。

表 41. 缺省值已更改的配置参数

参数	V8.2 缺省值	V9.1 缺省值
app_ctl_heap_sz － 应用程序控制堆大小配置参数	带有本地和远程客户机的数据库服务器：128 带有本地客户机的数据库服务器： • 64（适用于非 UNIX 平台） • 128（适用于基于 UNIX 的平台） 带有本地和远程客户机的分区数据库服务器：512	带有本地和远程客户机的数据库服务器： • 未启用 INTRA_PARALLEL 时：128 • 启用了 INTRA_PARALLEL 时：512 带有本地客户机的数据库服务器： • 未启用 INTRA_PARALLEL 时：64（适用于非 UNIX 平台） • 启用了 INTRA_PARALLEL 时：512（适用于非 UNIX 平台） • 未启用 INTRA_PARALLEL 时：128（适用于基于 UNIX 的平台） • 启用了 INTRA_PARALLEL 时：512（适用于基于 UNIX 的平台） 带有本地和远程客户机的分区数据库服务器：512
auto_maint － 自动维护配置参数	OFF	ON
auto_runstats（有关详细信息，请参阅 auto_maint － 自动维护配置参数。）	OFF	ON
auto_tbl_maint（有关详细信息，请参阅 auto_maint － 自动维护配置参数。）	OFF	ON
avg_appls － 平均活动应用程序数配置参数	1	自动
database_memory － 数据库共享内存大小配置参数	自动	• AIX 和 Windows：自动 • Linux、HP-UX、Solaris 操作系统：已计算
java_heap_sz － 最大 Java 解释器堆大小配置参数	512	• 32 位平台：512 • 64 位平台：1024

表 41. 缺省值已更改的配置参数（续）

参数	V8.2 缺省值	V9.1 缺省值
locklist － 锁定列表的最大存储器配置参数	• UNIX: 100 • 带有本地和远程客户机的 Windows 数据库服务器: 50 • 带有本地客户机的 Windows 64 位数据库服务器: 50 • 带有本地客户机的 Windows 32 位数据库服务器: 25	自动
maxlocks － 升级前锁定列表的最大百分比配置参数	• UNIX: 10 • Windows: 22	自动
num_iocleaners － 异步页清除程序的数目配置参数	1	自动
num_ioservers － I/O 服务器数目配置参数	3	自动
pckcachesz － 程序包高速缓存大小配置参数	-1	自动
sheapthres － 排序堆阈值配置参数	• UNIX 32 位平台: 20 000 • Windows 32 位平台: 10 000 • 64 位平台: 20 000	0
sheapthres_shr － 共享排序的排序堆阈值配置参数	*sheapthres*	自动
sortheap － 排序堆大小配置参数	256	自动
userexit － 启用用户出口配置参数	否	关闭

配置参数更改:

Windows	UNIX

更改:

下列配置参数在版本 9 中不再有效:

- *estore_seg_sz*
- *num_estore_segs*
- *min_priv_mem*（仅适用于 Windows）
- *priv_mem_thresh*（仅适用于 Windows）
- *fcm_num_rqb*
- *fcm_num_anchors*
- *fcm_num_connect*

已经对配置参数的内容和含义进行了下列更改:

- *avg_appls*；平均活动应用程序数数据库配置参数具有新的缺省值。除非检测到 SAP 环境，否则，该配置参数的缺省值设置为 1（旧缺省值）。如果是 SAP 环境，则缺省值为 3 个活动应用程序。在这两种情况下，设置的缺省值可能都不会与数据库配置参数 *maxappls* 的设置发生冲突。

  新格式按一种便于阅读的形式提供了端口号和 IP 地址，并且还接受更长的 IPv6 地址。

  如果您具有用来对包含应用程序标识的输出进行解析的脚本，则需要修改解析条件以说明新格式。例如，可以解析 LIST APPLICATIONS 命令的输出。
- *database_memory*；"AUTOMATIC"的含义已更改。在版本 9 之前称为"AUTOMATIC"，而在版本 9 中已重命名为"COMPUTED"。要使此配置参数的行为与 版本 9 之前一样，应将配置参数 *database_memory* 设置为"COMPUTED"。通过在版本 9 中将配置参数 *database_memory* 设置为"AUTOMATIC"，就可以启用内存自调整，并且会自动调整数据库内存的总使用量。
- *sheapthres_shr*；使用共享内存的排序类型已更改。在版本 9 之前，只有在对称多处理器（SMP）环境中进行排序或者在使用集中器时进行排序时才会使用共享内存。在版本 9 中，通过将 *sheapthres* 实例配置参数设置为零（0），并将 *sheapthres_shr* 数据库配置参数设置为非零值，数据库的所有排序内存使用者都将使用数据库共享内存而不使用专用排序内存。此外，*sheapthres_shr* 数据库配置参数的缺省值也从 *sheapthres* 值更改为 5000 4 KB 页。
- *dyn_query_mgmt*；在从版本 8 迁移至版本 9 期间，此配置参数的缺省值从"启用"更改为"禁用"。在完成迁移并且安装了 Query Patroller 之后，必须手工将此配置参数设置为"启用"。可以使用 UPDATE DATABASE CONFIGURATION 命令来设置配置参数值。
- *num_iocleaners* 和 *num_ioservers*；为这两个配置参数设置的缺省值已更改为"AUTOMATIC"。这表示启动的预取程序和页清除程序数基于环境特征（例如 CPU 数、数据库分区数以及数据库表空间并行性设置）。

  对于现有数据库来说，可以通过将 *num_iocleaners* 和 *num_ioservers* 设置为"AUTOMATIC"来利用此功能。

已对注册表变量的内容和含义进行了下列更改:

- DB2_ALLOCATION_SIZE；缺省值已从 8388608 更改为 131072。此注册表变量指定为缓冲池分配的内存大小。
- DB2_FORCE_FCM_BP；缺省值已从"否"更改为"是"。此注册表变量指定为 FCM 缓冲区分配的内存。

相关概念:

- 『DB2 服务器的迁移概述』（*《迁移指南》*）
- 『DB2 服务器的迁移建议』（*《迁移指南》*）
- 『DB2 客户机的迁移要点』（*《迁移指南》*）
- 『关于发行说明』（*《发行说明》*）
- 『V9.1 新增内容：管理更改总结』（*《新增内容》*）
- 『V9.1 新增内容：应用程序开发更改总结』（*《新增内容》*）
- 『V9.1 新增内容：现有功能的更改总结』（*《新增内容》*）
- 『V9.1 新增内容：数据库设置更改总结』（*《新增内容》*）

相关参考:

- 第 227 页的『不推荐使用和不继续使用的功能』

版本 8 与先前发行版之间的不兼容性

系统目录信息:

目录表中的 IMPLEMENTED 列:

Windows	UNIX

更改:

在先前版本中，SYSIBM.SYSFUNCTIONS 和 SYSCAT.SYSFUNCTIONS 中的 IMPLEMENTED 列具有值 Y、M、H 和 N。在版本 8 中，值是 Y 和 N。

解决方案:

对您的应用程序编码以仅使用值 Y 和 N。

OBJCAT 视图重命名为 SYSCAT 视图:

Windows	UNIX

更改:

已经将下列 OBJCAT 视图重命名为 SYSCAT 视图：TRANSFORMS、INDEXEXTENSIONS、INDEXEXTENSIONMETHODS、INDEXEXTENSIONDEP、INDEXEXTENSIONPARMS、PREDICATESPECS 和 INDEXEXPLOITRULES。

解决方案:

对您的应用程序编码以使用 SYSCAT 视图。

SYSCAT 视图现在是只读的:

Windows	UNIX

更改:

对于版本 8，SYSCAT 视图是只读的。

症状:

对 SYSCAT 模式中的视图的 UPDATE 或 INSERT 操作现在失败。

解释:

建议使用 SYSSTAT 视图来更新系统目录信息。某些 SYSCAT 视图可能在无意中被设置为可更新，现在已得到修正。

解决方案:

更改应用程序来引用可更新的 SYSSTAT 视图作为替代。

应用程序编程:

审计上下文记录语句大小已增长:

Windows	UNIX

更改:

语句限制已提高为 2 MB。

症状:

审计上下文记录语句文本太大，以致于表中容纳不下。

解释:

用来记录审计上下文记录的现有表只允许语句文本的大小为 32 KB。新的语句限制为 2 MB。如果不使用长语句，这将对您没有影响。

解决方案:

创建新表以保存审计上下文记录（将 CLOB(2M) 值用于语句文本列）。如果期望的话，则将旧表中的数据植入新表，然后删除旧表并使用新表。可以将新表重命名为与旧表的名称相同。重新绑定使用新表的任何应用程序。

缺省情况下应用程序以多线程方式运行:

Windows	UNIX

更改:

在版本 8 中，缺省情况下，应用程序是以多线程方式运行的。在先前版本中，缺省情况下是以单线程方式运行应用程序的。此更改意味着对 sqleSetTypeCtx API 的调用将不起作用。

版本 8 多线程方式等价于使用版本 8 之前的应用程序中的 SQL_CTX_MULTI_MANUAL 选项来调用 sqleSetTypeCtx API。版本 7 客户机仍然可以单线程方式运行应用程序。

解释:

在版本 7 中，如果想以多线程方式运行应用程序，则必须调用上下文 API 并管理上下文。在版本 8 中，不需要这样做，原因是 DB2 数据库 Linux 版、UNIX 版和 Windows 版将在内部管理上下文。但是，在版本 8 中，您仍然能够通过外部上下文 API 来管理应用程序的上下文（如果您想这样做的话）。

使用 VERSION 选项时，未返回 SQL0818N 错误:

Windows	UNIX

更改:

如果在 PRECOMPILE、BIND、REBIND 和 DROP PACKAGE 命令上使用 VERSION 选项，则要执行的请求现在可能返回 SQL0805N 错误而不是 SQL0818N 错误。

症状:

编码为对 SQL0818N 错误作出反应的应用程序的行为可能与以前不同。

解决方案:

将应用程序重新编码以对 SQL0805N 和 SQL0818N 错误作出反应。

未定义主变量时，不会对预编译器返回 SQL0306N 错误:

Windows	UNIX

更改:

如果未在 BEGIN DECLARE 部分声明主变量并且在 EXEC SQL 部分中使用它，则预编译器不会返回 SQL0306N。如果变量是在应用程序中的其他位置声明的，则应用程序运行时将返回 SQL0804N。如果未在应用程序中的任何位置声明该变量，则编译器会在编译时返回错误。

症状:

编码为在预编译时对 SQL0306N 错误作出反应的应用程序的行为可能与以前不同。

解决方案:

应在 BEGIN DECLARE 部分声明主变量。如果主变量是在不同于 BEGIN DECLARE 部分的部分中声明的，则应将应用程序重新编码以处理 SQL0804 返回码。

与可滚动游标配合使用时不受支持的数据类型:

Windows	UNIX

更改:

在版本 8 中，不支持使用 LONG VARCHAR、LONG VARGRAPHIC、DATALINK 和 LOB 类型以及上述任何一种类型的单值类型或结构类型的可滚动游标。版本 7 的可滚动游标支持的上述数据类型将不再受支持。

症状:

如果在可滚动游标的选择列表中指定了具有这些数据类型的任何列，将返回 SQL0270N 原因码 53。

解决方案:

修改可滚动游标的选择列表，以便该列表不包括具有上述任何一种类型的列。

代码页转换表的欧洲版本:

Windows	UNIX

更改:

版本 8 代码页转换表提供对欧元符号的支持，这与随 DB2 的先前版本提供的转换表稍有不同。

解决方案:

如果想要使用版本 8 之前的代码页转换表，它们在目录 `sqllib/conv/v7` 中。

在 LOB 定位器和 LOB 值之间切换:

Windows	UNIX

更改:

在游标语句的绑出期间已经更改了大对象（LOB）定位器和 LOB 值之间切换的功能。当应用程序与 SQLRULES DB2 绑定在一起时（缺省行为），用户将不能在 LOB 定位器和 LOB 值之间切换。

解决方案:

如果想要在游标语句的绑出期间在 LOB 定位器和 LOB 值之间切换，则使用 SQLRULES STD 预编译应用程序。

UNIX 平台上的未落实工作单元:

	UNIX

更改:

在版本 8 中，所有应用程序终止都隐式地回滚未完成的工作单元。基于 Windows 的应用程序将不会更改，因为它们已经对正常或异常应用程序终止执行了隐式回滚。在版本 8 之前，如果未使用显式或隐式上下文支持的 UNIX 应用程序正常终止而不直接调用 CONNECT RESET、COMMIT 或 ROLLBACK 语句，则该应用程序将落实未完成的工作单元。如果 CLI、ODBC 和基于 Java 的应用程序（隐式上下文支持）和显式创建应用程序上下文的应用程序终止，则它们总是回滚未完成的工作单元。异常应用程序终止还将导致未完成工作单元的隐式回滚（ROLLBACK）。

解决方案:

为了确保落实事务，应用程序在终止前应执行显式 COMMIT 或 CONNECT RESET。

对保存点命名的更改:

Windows	UNIX

更改:

保存点名称不能再以“SYS”开始。

症状:

以“SYS”开始的名称创建保存点将会失败，返回错误 SQL0707N。

解释:

以“SYS”开始的保存点名称是保留为系统使用的。

解决方案:

将以“SYS”开始的所有保存点重命名为不以“SYS”开始的另一名称。

代码页转换错误和字节替换:

Windows	UNIX

更改:

在有需要的时候，将把输入主变量中的字符数据转换到数据库代码页，然后才在其中出现该主变量的 SQL 语句中使用该字符数据。在代码页转换期间，可能会发生数据扩充。以前，当对主变量中的数据检测到代码页转换时，将增大对该主变量假定的实际长度以处理该扩充。现在不再执行这种假定的长度增加，以减轻数据类型长度的更改对其他 SQL 操作的影响。

注: 这些情况都不适用于在 FOR BIT DATA 的上下文中使用的主变量。在将这些主变量中的数据用作位数据之前，不对它们进行转换。

症状:

如果主变量不够大，从而无法存放代码页转换之后的扩充长度，则返回错误（SQLSTATE 22001，SQLCODE -302）。

解释:

因为扩展或压缩可能在代码页转换期间发生，取决于主变量中数据长度的操作可能产生不同的结果或错误情况。

解决方案:

可以考虑的备用方法包括:

- 将应用程序编码为通过增加字符主变量的长度来处理导致数据长度发生更改的代码页转换的可能性
- 更改数据以避免导致扩充的字符
- 将应用程序代码页更改为与数据库代码页相匹配，以便不发生代码页转换。

主变量的代码页转换:

Windows	UNIX

更改:

现在代码页转换将在绑定阶段执行（如果需要的话）。

症状:

不同的结果。

解释:

既然总是会执行主变量的代码页转换（必要时），谓词求值将总是在数据库代码页中进行，而不是在应用程序代码页中进行。例如，

```
SELECT * FROM table WHERE :hv1 > :hv2
```

将使用数据库代码页执行，而不是应用程序代码页。使用的整理仍然是数据库整理。

解决方案:

验证先前版本中的结果是否为真正想要的结果。如果答案是肯定的，则在使用数据库整理和代码页时，更改谓词以产生想要的结果。或者更改应用程序代码页或数据库代码页。

主变量中数据的扩展和压缩:

Windows	UNIX

更改:

现在代码页转换将在绑定操作期间执行（如果需要的话）。

症状:

来自主变量的数据的长度不同。

解释:

因为扩展或压缩可能在代码页转换期间发生，取决于主变量中数据长度的操作可能产生不同的结果或错误情况。

解决方案:

更改数据、应用程序代码页或数据库代码页，以使代码页转换不产生转换的数据长度的更改，或编码应用程序以处理代码页转换可能导致的数据长度更改。

代码页转换后主变量的长度:

Windows	UNIX

更改:

代码页转换将不再因为扩充而导致主变量或参数标记符的长度增加。

症状:

数据截断错误。

解释:

不再因为代码页转换的潜在扩充而增加对隐式类型参数标记确定的字符数据类型的长度。对于使用隐式类型参数标记的长度确定结果长度的操作，结果长度将更短。例如，如果 C1 是 CHAR(10) 列:

```
VALUES CONCAT (?, C1)
```

对于从应用程序代码页向数据库代码页转换时可以扩充 3 倍的数据库，结果数据类型和长度将不再是 CHAR(40)，而是结果数据类型和长度为 CHAR(20)。

解决方案:

使用 CAST 以对隐式类型参数标记指定想要的类型，或将确定隐式类型参数标记的类型的操作数更改为可容纳因代码页转换而造成的数据扩充的数据类型或长度。

对 DESCRIBE 语句的输出的更改:

Windows	UNIX

更改:

代码页转换将不再因为扩充而导致主变量或参数标记符的长度增加。

症状:

来自 DESCRIBE 语句的输出将会更改。

解释:

因为结果长度不会因代码页转换的潜在扩充而增加，描述这一结果长度的 DESCRIBE 语句的输出现在将会不同。

解决方案:

必要时更改应用程序以处理 DESCRIBE 语句返回的新值。

将 SUBSTR 函数与主变量配合使用时出错:

Windows	UNIX

更改:

代码页转换将不再因为扩充而导致主变量或参数标记符的长度增加。

症状:

SUBSTR 导致错误 SQL0138N。

解释:

考虑由于代码页转换导致的潜在扩充，方法是增加为主变量保留的长度。这是允许的，例如，对长度为 10 的主变量成功执行的 SUBSTR（:hv,19,1）。它将不再有效。

解决方案:

增加主变量的长度以备转换数据的长度所用，或更改 SUBSTR 调用以指定主变量长度内的位置。

Solaris 不再支持非线程安全库:

	UNIX

更改:

非线程安全库 libdb2_noth.so 不再可用。

症状:

需要 libdb2_noth.so 的工具或应用程序将不再有效。

解释:

因为不再需要支持废弃的非线程安全库，所以 IBM DB2 Solaris 版版本 9.1 不包含 libdb2_noth.so 库。

解决方案:

更改工具或应用程序，以使用线程安全的 libdb2.so 库。使用 -mt 参数重新链接应用程序。

连接至 Unicode 数据库时导入或导出 DBCLOB:

Windows	UNIX

更改:

在版本 8 之前，如果导出的数据中包含来自 Unicode 数据库（UTF-8）的 DBCLOB，并且使用了 LOBSINFILE 文件类型修饰符，则将在代码页 1200（Unicode 图形代码页）中导出 DBCLOB。如果导入的数据包含 DBCLOB，并且使用了 LOBSINFILE 文件类型修饰符，则将在代码页 1200（Unicode 图形代码页）中导入 DBCLOB。如果将 DB2GRAPHICUNICODESERVER 注册表变量设置为 ON，则版本 8 中将保持此行为。

在版本 8 中，DB2GRAPHICUNICODESERVER 注册表变量的缺省设置为 OFF。如果导出的数据包含 DBCLOB，并使用 LOBSINFILE 文件类型修饰符，则将在应用程序的图形代码页中导出 DBCLOB。如果导入的数据包含 DBCLOB，并使用 LOBSINFILE 文件类型修饰符，则将在应用程序的图形代码页中导入 DBCLOB。如果应用程序代码页是 IBM-eucJP（954）或 IBM-eucTW（964），并且导出的数据包含 DBCLOB，并使用

LOBSINFILE 文件类型修饰符，则将以应用程序的字符代码页导出 DBCLOB。如果导入的数据包含 DBCLOB，并使用 LOBSINFILE 文件类型修饰符，则将以应用程序的字符代码页导入 DBCLOB。

症状:

将带有 LOBSINFILE 文件类型修饰符的数据导入 Unicode 数据库时，将正确地转换字符数据，但是 DBCLOB 数据会被毁坏。

解决方案:

如果您要在版本 8 数据库和较早版本的数据库之间移动数据，则将 DB2GRAPHICUNICODESERVER 注册表变量设置为 ON 以保留先前的行为。

***SQL*:**

CREATE TABLESPACE 语句的 DROPPED TABLE RECOVERY 缺省值已更改:

Windows	UNIX

更改:

CREATE TABLESPACE 的 DROPPED TABLE RECOVERY 缺省值已从 OFF 更改为 ON。

症状:

缺省值的这种更改可能会影响正向恢复性能。当有许多废弃表操作要恢复时，或者当历史记录表较大时，对性能的影响可能很大。在后一种情况下，在正向恢复期间重做 SQLP_PRT_DROP_TABLE_RECOVERY 暂挂列表操作时，数据库管理器需要读取并更新相关已废弃的恢复条目的历史记录文件。如果历史记录文件较大，则此操作在搜索该文件以找到正确的条目并更新该条目时可能要花一些时间。

解决方案:

如果将废弃许多表，并且如果要使用循环日志记录或不想恢复这些表，则应该考虑禁用此功能。要禁用新表空间的此功能，可使用 CREATE TABLESPACE 语句并显式使用值设置为 OFF 的 DROPPED TABLE RECOVERY 子句。对于现有表空间，可使用 ALTER TABLESPACE 语句并显式使用 DROPPED TABLE RECOVERY 子句，以将缺省值（ON）更改为 OFF。已废弃的表将不再作为正向恢复过程的一部分来恢复。

不允许函数和过程有相同特定名称:

Windows	UNIX

更改:

已统一 SPECIFICNAME 的名称空间。先前版本的 DB2 将允许函数和过程具有相同的特定名称，但是版本 8 不允许这样做。

症状:

如果在将数据库迁移至版本 8，则 db2ckmig 实用程序将检查函数和过程是否具有相同特定名称。如果在迁移过程中遇到重复的名称，则迁移将会失败。

解决方案:

删除过程并以另一特定名称重新创建该过程。

对函数和过程的 EXECUTE 特权:

Windows	UNIX

更改:

以前，用户只需创建例程以使其他用户能够使用它。现在，在创建例程后，用户必须对其授予 EXECUTE 权限，其他用户才能使用它。

在先前版本中，对于过程没有权限检查，但是调用者必须具有对从该过程调用的任何程序包具有 EXECUTE 特权。在版本 8 中，对于使用 CALL_RESOLUTION IMMEDIATE 预编译的嵌入式应用程序，以及对于 CLI 编目的过程，调用者必须具有对过程的 EXECUTE 特权，而只有该过程的定义者才必须具有对所有程序包的 EXECUTE 特权。

症状:

1. 应用程序可能不能正常工作。
2. 由多个程序包组成的现有过程（并且过程的定义者对其不具有对所有程序包的访问权）将不能正常工作。

解决方案:

1. 发出必需的 GRANT EXECUTE 语句。如果所有的例程都在单个模式中，则可以使用单个语句授予对每种类型的例程的特权，例如:

   ```
   GRANT EXECUTE ON FUNCTION schema1.* TO PUBLIC
   ```

2. 如果一个程序包对每个用户都可用，但是另一程序包被限制为只有几个有特权的用户可以使用，则使用这两个程序包的存储过程在尝试访问第二个程序包时，将会发现有在权限错误。如果它看到权限错误，则它将知道该用户不是特权用户并且该过程将绕过它的逻辑部分。

   可用以下几种方法解决此情况:

   a. 当对程序进行预编译时，应设置 CALL_RESOLUTION DEFERRED 以指示当预编译器无法解析 CALL 语句上的过程时，程序将作为废弃的 sqleproc() API 的调用来执行。
   b. 可以将 CLI 关键字 UseOldStpCall 添加至 db2cli.ini 文件以控制调用过程的方式。它可以有两个值：值 *0* 意味着将不会使用旧的调用方法来调用过程，而值 *1* 意味将使用旧的调用方法调用过程。
   c. 对执行程序包的所有用户授予 EXECUTE 特权。

对表添加外键约束:

Windows	UNIX

更改:

在先前版本中，如果创建引用处于检查暂挂状态的表的外键约束，则还将使从属表处于检查暂挂状态。在版本 8 中，如果创建引用处于检查暂挂状态的表的外键约束，则有两种可能的结果:

1. 如果在创建从属表时添加外键约束，则表的创建和约束的添加操作将会成功，因为表创建时将为空，所以没有违反约束的行。
2. 如果向现有表添加外键，则将接收到错误 SQL0668N。

解决方案:

在添加引用表的外键前，使用 SET INTEGRITY ... IMMEDIATE CHECKED 语句来打开对于处于检查暂挂状态的表的完整性检查。

对 SET INTEGRITY ... IMMEDIATE CHECKED 的更改:

Windows	UNIX

更改:

在先前发行版中，对其发出 SET INTEGRITY ... UNCHECKED 语句的表（即，SYSCAT.TABLES 的 const_checked 列带有一些 “U” 字节）缺省情况下将在执行下一个 SET INTEGRITY ... IMMEDIATE CHECKED 语句时得到完全处理，这意味着将检查所有记录的约束违例情况。必须显式指定 INCREMENTAL 以避免完全处理。

在版本 8 中，当发出 SET INTEGRITY ... IMMEDIATE CHECKED 语句时，缺省情况是仅进行增量处理，而让未检查数据保持原样（即，保留 “U” 字节）。（将返回警告，指示旧数据保持未验证状态。）

解释:

进行此更改以避免缺省行为（对所有记录执行约束检查），这通常会消耗较多的资源。

解决方案:

您将必须显式指定 NOT INCREMENTAL 以强制执行完全处理。

CHAR 函数的十进制分隔符:

Windows	UNIX

更改:

在语言环境使用逗号作为十进制分隔符且 CHAR 函数的非限定调用包括类型为 REAL 或 DOUBLE 的自变量的服务器上运行的动态应用程序将在 CHAR(double) 函数的结果中返回句号作为分隔符。在版本 8 中重新创建像视图和触发器这样的对象或在显式重新绑定静态程序包时，也会出现这种不兼容性。

解释:

这是解析新 SYSIBM.CHAR(double) 函数特征符而不是 SYSFUN.CHAR(double) 特征符的结果。

解决方案:

要维护来自较早版本的 DB2 的行为，应用程序将需要使用 SYSFUN.CHAR 显式调用函数而不是允许函数解析选择 SYSIBM.CHAR 特征符。

对 CALL 语句的更改:

Windows	UNIX

更改:

在版本 8 中，使用 CALL_RESOLUTION IMMEDIATE 预编译的应用程序和 CLI 编目的过程与先前版本相比较有几个关键不同之处:

- 主变量支持已替换为对动态 CALL 的支持。
- 已除去对调用未编目存储过程的应用程序的编译的支持。在 DB2 的将来版本中，将完全除去对未编目存储过程的支持。
- 已经取消对可变自变量列表存储过程的支持。
- 装入存储过程库有几个不同的规则。

解决方案:

在版本 8 之前受支持的 CALL 语句仍然可用，并且可以在 PRECOMPILE PROGRAM 命令上使用 CALL_RESOLUTION DEFERRED 选项来进行访问。

现有应用程序（在版本 8 之前构建的）将继续工作。如果重新预编译应用程序时未使用 CALL_RESOLUTION DEFERRED 选项，则可能需要更改源代码。

在将来的版本中，将除去对 CALL_RESOLUTION DEFERRED 语句的支持。

来自 UDF 的输出返回定长字符串:

Windows	UNIX

更改:

可将 UDF（标量或表函数）定义为返回定长字符串（CHAR(n) 或 GRAPHIC(n)）。在先前版本中，如果返回的值包含嵌入式空字符，则结果将仅为 n 个字节（而对于 GRAPHIC 数据类型为 2n 个字节），包括空字符和空字符右边的所有字节。在版本 8 中，DB2 UDB 查找空字符并从该位置（空字符）到值的结束返回空格。

解决方案:

如果想要继续版本 8 之前的行为，则将 CHAR(n) 返回的值的定义更改为 CHAR(n) FOR BIT DATA。对于 GRAPHIC 数据，没有任何方法可继续执行版本 8 之前的行为。

数据库连接行为中的更改:

Windows	UNIX

更改:

在版本 7 中，如果使用嵌入式 SQL 连接至数据库，然后试图连接至不存在的数据库，则试图连接至不存在的数据库将会失败，返回 SQL1013N。与第一个数据库的连接仍然存在。在版本 8 中，试图连接至不存在的数据库将导致与第一个数据库断开连接。这将导致应用程序处于没有任何连接的状态。

解决方案:

编码嵌入式 SQL 以在与另一数据库的连接尝试不成功后重新连接至初始数据库。

撤销对程序包的 CONTROL 特权:

Windows	UNIX

更改:

用户可以使用 CONTROL 特权授予对程序包的特权。DB2 UDB 版本 8 中，WITH GRANT OPTION 提供了一种机制来确定用户将对程序包的特权授予其他用户的权限。使用此机制代替 CONTROL 以确定用户是否能够将特权授予其他用户。撤销 CONTROL 特权时，用户将仍然能够将特权授予其他用户。

症状:

用户在撤销 CONTROL 特权之后，仍能授予对程序包的特权。

解决方案:

如果用户应不再有权将对程序包的特权授予其他用户，则撤销对程序包的所有特权并仅授予必需的特权。

将 FOR BIT DATA 字符串数据类型转换为 CLOB 时出错:

Windows	UNIX

更改:

将定义为 FOR BIT DATA 的字符串数据类型转换为 CLOB（使用 CAST 规范或 CLOB 函数）现在返回错误（SQLSTATE 42846）。

症状:

数据类型转换为 CLOB 现在返回错误，而先前不会。

解释:

CLOB 数据类型不支持 FOR BIT DATA。未定义在将 FOR BIT DATA 字符串指定为自变量时使用 CAST 规范或 CLOB 函数的结果。此情况现在作为错误捕获。

解决方案:

将自变量更改为 CAST 规范或 CLOB 函数，这样它就不是 FOR BIT DATA 字符串。可通过使用 CAST 规范将 FOR BIT DATA 字符串强制类型转换为 FOR SBCS DATA

字符串或 FOR MIXED DATA 字符串来完成此操作。例如，在非 DBCS 数据库中，若 C1FBD 是声明为 FOR BIT DATA 的 VARCHAR(20) 列，则以下将是 CLOB 函数的有效自变量:

```
CAST (C1FBD AS VARCHAR(20) FOR SBCS DATA)
```

来自 CHR 函数的输出:

Windows	UNIX

更改:

对于代码点 X'00'，CHR(0) 将返回空白（X'20'）而不是字符。

症状:

如果将 X'00' 作为自变量，则 CHR 函数的输出将返回不同的结果。

解释:

当从用户定义的函数调用和返回时，字符串处理将把 X'00' 解释为字符串结束。

解决方案:

更改应用程序代码以处理新的输出值。或者，定义一个用户定义的函数，它返回来源于 SYSFUN CHR 函数定义的 CHAR(1) FOR BIT DATA，并在 SQL 路径上将此函数置于 SYSFUN 前面。

例如，要找到位于 IMPLEMENTATION 列中的 SYSFUN.CHR 的源定义:

```
SELECT IMPLEMENTATION, ROUTINENAME FROM SYSIBM.SYSROUTINES
   WHERE ROUTINENAME LIKE '%CHR%';

IMPLEMENTATION          ROUTINENAME
--------------          -----------
db2clifn!CLI_udfCHAR    CHR
```

然后，可从上面返回的定义 *db2clifn!CLI_udfCHAR* 创建一个新的用户定义的函数。

```
CREATE FUNCTION DBS.CHR(INTEGER) RETURNS CHARACTER(1) FOR BIT DATA NOT FENCED
   LANGUAGE C PARAMETER STYLE DB2SQL NO DBINFO EXTERNAL NAME 'db2clifn!CLI_udfCHAR'
```

无法在生成列或检查约束中使用 TABLE_NAME 和 TABLE_SCHEMA 函数:

Windows	UNIX

更改:

已更正 TABLE_NAME 和 TABLE_SCHEMA 函数的定义，现在无法在生成列或检查约束中使用这些函数。

症状:

绑定将会失败，且 SQLCODE -548/SQLSTATE 42621 声明 TABLE_NAME 或 TABLE_SCHEMA 在检查约束的上下文中无效。

解释:

TABLE_NAME 和 TABLE_SCHEMA 函数从目录视图中检索数据。它们属于 READS SQL DATA 类；因为 DB2 不能随时间的推移强制执行约束更正，所以 GENERATED COLUMN 表达式和检查约束中不允许 READS SQL DATA 类函数。

解决方案:

更新包含生成列表达式和检查约束的所有列，以便不再使用 TABLE_NAME 和 TABLE_SCHEMA。要改变生成列，可使用 ALTER TABLE 语句来“设置”新表达式。要除去检查约束，使用带有 DROP CONSTRAINT 子句的 ALTER TABLE 语句。这将允许您“绑定”并继续访问包含受影响列的表。

数据库安全性和调整:

对 CREATE FUNCTION、CREATE METHOD 和 CREATE PROCEDURE 语句的权限:

Windows	UNIX

更改:

在版本 8 中引入了 CREATE_EXTERNAL_ROUTINE 权限。

症状:

带有 EXTERNAL 选项的 CREATE FUNCTION、CREATE METHOD 和 CREATE PROCEDURE 语句可能会失败。

解决方案:

对发出带有 EXTERNAL 选项的 CREATE FUNCTION、CREATE METHOD 和 CREATE PROCEDURE 语句的用户授予 Grant CREATE_EXTERNAL_ROUTINE 权限。

实用程序和工具:

对 DB2 管理服务器（DAS）的更改:

Windows	UNIX

症状:

迁移时，发生了错误且在 DB2 启动时已将这些错误记录到 db2diag.log 中。

解释:

在版本 7 中，DB2 管理服务器（DAS）是其本身的实例。在版本 8 中，DAS 不再是一个实例，而是一个用于协助 DB2 服务器任务的控制点。

解决方案:

如果您已尝试从版本 7 迁移到版本 8，并遇到与 DAS 相关的问题，则使用“db2admin drop”命令来停止和删除 DAS。在除去 DAS 之后，接着使用“db2admin create”命令来创建 DAS。

使用控制中心监视性能时的更改:

Windows	UNIX

症状:

当在控制中心内查找时，找不到对性能监视器的任何引用。

解释:

已经除去了控制中心的性能监视功能。

解决方案:

当使用 IBM DB2 Windows 版版本 9.1 时，可以使用下列工具来监视性能:

- **DB2 性能专家**

  单独购买的用于多平台的 DB2 性能专家 V1.1 根据与 DB2 性能相关的信息来合并、报告、分析和建议自管理和资源调整更改。

- **DB2 运行状况中心**

  运行状况中心的功能使您能够通过不同方法来处理与性能相关的信息。这些功能在某些方面替代了控制中心的性能监视器功能。

- **Windows 性能监视器**

  “Windows 性能监视器”使您能够监视数据库和系统性能，可从向系统注册的任何性能数据提供程序检索信息。Windows 还提供有关机器运行的所有方面的性能信息数据，包括下列方面:

  – CPU 使用情况
  – 内存使用率
  – 磁盘活动
  – 网络活动

同时运行联机实用程序:

Windows	UNIX

症状:

当同时使用联机实用程序时，这些实用程序可能要花很长时间才能完成。

解释:

每个实用程序需要的锁定都会影响同时运行的其他实用程序的进度。

解决方案:

当同时运行的各实用程序的锁定需求之间存在潜在冲突时，应该考虑改变您想运行的实用程序的调度。实用程序（例如，联机备份表空间、装入表或表的适当位置重组）使用锁定机制来防止实用程序之间发生冲突。实用程序使用表锁定、表空间锁定和不同时间的表空间状态来控制需要在数据库中完成的操作。当一个实用程序挂起锁定时，请求相似或相关锁定的其他实用程序就必须一直等待，直到这些锁定被释放为止。

例如，当包括正在重组的表的联机备份正在运行时，就不能开始执行适当位置表重组的最后一个阶段。如果需要完成备份，则可以暂停重组请求。

在另一个示例中，联机 Load 实用程序将不会处理对同一个表的另一个联机装入请求。如果正在装入不同的表，则装入请求将不会相互阻塞。

对 db2move 总结输出的更改:

Windows	UNIX

更改:

在版本 8.2 中，**db2move** 生成的总结输出的描述性更强。但是，总结输出中的更改可能会导致编写的用来分析旧输出的脚本失败。

症状:

编写的用来分析 **db2move** 生成的旧输出的脚本失败。

解释:

改进了 **db2move** 生成的总结输出。

当带“IMPORT”选项运行 **db2move** 时，旧输出显示为:

```
IMPORT: -Rows read:     5; -Rows committed:     5;   Table "DSCIARA2"."T20"
```

新输出显示为:

```
   * IMPORT:  table "DSCIARA2"."T20"
  -Rows read:         5
  -Inserted:          4
  -Rejected:          1
  -Committed:         5
```

当带“LOAD”选项运行 **db2move** 时，旧输出显示为:

```
* LOAD:  table "DSCIARA2"."T20"
  -Rows read:    5; -Loaded:    4;  -Rejected   1  -Deleted   0  -Committed   5
```

新输出显示为:

```
   * IMPORT:  table "DSCIARA2"."T20"
  -Rows read:         5
  -Loaded:            4
  -Rejected:          1
  -Deleted:           0
  -Committed:         5
```

解决方案:

将需要修改用来分析 **db2move** 输出的脚本，以考虑布局和内容的更改。

对说明工具表的更改:

Windows	UNIX

更改:

在版本 8 中，对现有说明工具表（还包括下面两个新表）进行了更改：ADVISE_MQT 和 ADVISE_PARTITION。

症状:

当要求对具体化查询表（MQT）或者数据库分区提供建议时，如果尚未创建说明表，则“DB2 设计顾问程序”将返回错误消息。

解释:

尚未创建新表 ADVISE_MQT 和 ADVISE_PARTITION。

解决方案:

使用 **db2exmig** 命令来将版本 7 和版本 8.1 说明表移至版本 8.2。此命令让必需的 EXPLAIN DLL 创建需要的所有说明工具表。

对 db2diag.log 消息格式的更改:

Windows	UNIX

更改:

在版本 8 中，更改了 db2diag.log 消息格式。

症状:

当您查看 db2diag.log 消息时，您将注意到格式已更改。这些更改包括下列示例：每条消息都将具有一个诊断日志记录头、记录字段前面将出现字段名和列，并且将清楚地标记出日志记录的消息和数据部分。对格式的所有更改都将使日志记录更容易使用和理解。

解释:

正在修改 DB2 诊断日志。db2diag.log 文件将是可解析的。

不支持低级 CREATE DATABASE 和 DROP DATABASE:

Windows	UNIX

更改:

在版本 8 中，不支持来自下级客户机或向下级服务器发出的 CREATE DATABASE 和 DROP DATABASE 命令。

症状:

发出其中一个命令时，将接收到错误 SQL0901N。

解释:

仅支持将 CREATE DATABASE 和 DROP DATABASE 命令从版本 8 客户机发送至版本 8 服务器。不能从版本 6 或版本 7 客户机向版本 8 服务器发出这些命令。不能从版本 8 客户机向版本 7 服务器发出这些命令。

解决方案:

从版本 8 客户机创建或删除版本 8 数据库。从版本 6 或版本 7 客户机创建或删除版本 7 数据库。

装入后对表的方式更改:

Windows	UNIX

更改:

在先前的版本中，对于使用 INSERT 选项装入且具有即时具体化查询表（又称为总结表）的表，在对其执行后继 SET INTEGRITY IMMEDIATE CHECKED 语句之后，该表将处于“常规”（完整访问）状态。在版本 8 中，表在执行 SET INTEGRITY IMMEDIATE CHECKED 语句后将处于“无数据移动”方式。

解释:

对处于“无数据移动”方式的表的访问与对处于“常规”（完整访问）方式的表的访问非常相似，但涉及该表内部的数据移动的一些语句和实用程序除外。

解决方案:

可以通过对基本表发出 SET INTEGRITY ... IMMEDIATE CHECKED FULL ACCESS 语句来强制已经装入且具有从属即时总结表的基本表绕过“无数据移动”方式并直接进入“完整访问”方式。但是，建议不要使用此选项，因为它将强制从属立即具体化查询表（又称为总结表）的完全刷新。

处于插入或替换方式的 Load 实用程序:

Windows	UNIX

更改:

在先前版本中，当使用处于插入或替换方式的 Load 实用程序时，若完整性检查关闭，则缺省选项是 CASCADE IMMEDIATE；当表置于检查暂挂状态时，还会立即将该表的所有从属外键表和从属具体化查询表（又称为总结表）置于检查暂挂状态。

对于版本 8，当使用处于插入或替换方式的 Load 实用程序时，如果已经关闭了完整性检查，则缺省值是 CASCADE DEFERRED。

解决方案:

可以使用 LOAD 命令的 CHECK PENDING CASCADE IMMEDIATE 选项来将从属外键表和从属具体化查询表以及它们的父表一起置于检查暂挂状态。

DB2_LIKE_VARCHAR 不会控制子元素统计信息的收集:

Windows	UNIX

更改:

在版本 7 中，DB2_LIKE_VARCHAR 注册表变量控制子元素统计信息的收集以及这些统计信息的使用。在版本 8 中，DB2_LIKE_VARCHAR 不会控制子元素统计信息的收集；然而，子元素统计信息的收集由 RUNSTATS 命令的 LIKE STATISTICS 选项或 db2Runstats API 的 iColumnflags 参数的 DB2RUNSTATS_COLUMN_LIKE_STATS 值控制。

症状:

在调用 RUNSTATS 命令或调用 db2Runstats API 之后，在系统目录中将子元素统计信息设置为 -1（缺省值）；可以通过如下所示的查询来观察这种情况:

```
SELECT SUBSTR(TABSCHEMA,1,18), SUBSTR(TABNAME,1,18),
   SUBSTR(COLNAME,1,18), COLCARD, AVGCOLLEN, SUB_COUNT, SUB_DELIM_LENGTH
   FROM SYSSTAT.COLUMNS
   WHERE COLNAME IN ('P_TYPE', 'P_NAME')
   ORDER BY 1,2,3
```

（将 P_TYPE 和 P_NAME 替换为适当的列名。）

如果对于 COLCARD 和 AVGCOLLEN 某列的结果具有非负值，而对于 SUB_COUNT 和 SUB_DELIM_LENGTH 该列的结果为 -1，这指示已经为该列收集了基本统计信息，但是尚未收集子元素统计信息。

解决方案:

如果在版本 7 中指定了 DB2_LIKE_VARCHAR=?,Y（其中 ? 是任何值），则应在 RUNSTATS 命令上指定 LIKE STATISTICS 选项或者在 db2Runstats API 上指定 DB2RUNSTATS_COLUMN_LIKE_STATS，以便为适当的列收集这些统计信息。

连接和共存:

先前级别服务器支持:

Windows	UNIX

更改:

在将环境从版本 7 移至版本 8 时，如果在将所有服务器迁移至版本 8 之前将客户机迁移至版本 8，则有几个限制。这些限制与 DB2 Connect 无关；也与 zSeries、OS/390 或 iSeries 数据库服务器无关。

解决方案:

为了使版本 8 客户机能够与版本 7 服务器一起工作，您需要在版本 7 服务器上配置／启用“DRDA 应用程序服务器”功能。有关如何完成此任务的信息，请参阅版本 7《安装与配置补充手册》。

为避免已知限制，在将任何客户机迁移至版本 8 之前，应将所有服务器迁移至版本 8。如果做不到这一点，则您应该知道，当从版本 8 客户机访问版本 7 服务器时，将不支持下列各项:

- 一些数据类型:
  - 大对象（LOB）数据类型。
  - 用户定义的单值类型（UDT）。
  - DATALINK 数据类型。
- 一些安全性功能:
  - 认证类型 SERVER_ENCRYPT。
  - 更改密码。您不能从 DB2 UDB 版本 8 客户机更改 DB2 UDB 版本 7 服务器上的密码。
- 某些连接和通信协议:
  - 需要 ATTACH 而不是需要连接的实例请求。
  - 不支持从 DB2 UDB 版本 8 客户机向 DB2 UDB 版本 7 服务器执行 ATTACH 语句。
  - 唯一受支持的网络协议是 TCP/IP。
  - 不支持其他网络协议，如 SNA、NetBIOS、IPX/SPX 及其他协议。
- 一些应用程序功能部件和任务:
  - 不支持 DESCRIBE INPUT 语句，但 ODBC/JDBC 应用程序例外。为支持运行 ODBC/JDBC 应用程序的 DB2 UDB 版本 8 客户机访问 DB2 UDB 版本 7 服务器，必须对所有需要此类型访问的 DB2 UDB 版本 7 服务器应用 DESCRIBE INPUT 支持修订。此修订与 APAR IY30655 相关联，将在“DB2 UDB 版本 8 一般可用性”日期之前提供。使用任何 DB2 文档中的“与 IBM 联系”信息来了解如何获取与 APAR IY30655 相关联的修订。DESCRIBE INPUT 语句是性能和可用性的增强，允许应用程序请求器在已准备语句中获取输入参数标记的描述。对于 CALL 语句，这包括与存储过程的 IN 和 INOUT 参数相关联的参数标记。
  - 使用 Result.getObject(1) 将返回 BigDecimal，而不是 JDBC 规范所要求的 Java Long 数据类型。当 DB2 UDB 版本 7 DRDA 服务器在响应 DESCRIBE INPUT 请求和检索数据时，它将 BIGINT 映射至 DEC(19,0)。出现这种行为的原因是 DB2 UDB 版本 7 服务器在 DRDA 级别上操作，但此级别没有定义 BIGINT。
  - 不支持查询中断。这影响 CLI/ODBC SQL_QUERY_TIMEOUT 连接属性以及中断 API。
  - 两阶段落实。当使用涉及 DB2 UDB 版本 8 客户机的协调事务时，不能将 DB2 UDB 版本 7 服务器用作事务管理器数据库。DB2 UDB 版本 7 服务器也不能参与 DB2 UDB 版本 8 服务器可能在其中充当事务管理器数据库的协调事务。
  - 符合 XA 的事务管理器。使用 DB2 UDB 版本 8 客户机的应用程序不能将 DB2 UDB 版本 7 服务器用作 XA 资源。这包括事务管理安排中的 WebSphere、Microsoft COM+/MTS、BEA WebLogic 和其他内容。
  - 监视。不支持从 DB2 UDB 版本 8 客户机向 DB2 UDB 版本 7 服务器执行监视功能。

– 实用程序。在下列情况下，不支持可以由客户机对服务器启动的那些实用程序:
  1. 客户机为 DB2 UDB 版本 8，而服务器为 DB2 UDB 版本 7。
  2. SQL 语句的大小大于 32 KB。
– 不支持查询中断。这影响 CLI/ODBC SQL_QUERY_TIMEOUT 连接属性以及中断 API。

除了对使用 DB2 UDB 版本 7 服务器的 DB2 UDB 版本 8 客户机的上述限制之外，对使用 DB2 UDB 版本 7 服务器的 DB2 UDB 版本 8 工具也存在类似的限制。下列 DB2 UDB 版本 8 工具只支持 DB2 UDB 版本 8 服务器:

- 控制中心
- 任务中心
- 日志
- 卫星管理中心
- 信息目录中心（包括此中心的 Web 版本）
- 运行状况中心（包括此中心的 Web 版本）
- 许可证中心
- Spatial Extender
- 工具设置
- 开发中心。您应该使用“存储过程构建器”来在版本 8 之前的服务器上开发服务器对象。

下列 DB2 UDB 版本 8 工具支持 DB2 UDB 版本 7 服务器（但有一些限制）和 DB2 UDB 版本 8 服务器:

- 配置助手

  发现 DB2 UDB 版本 7 服务器并对其进行编目是有可能的。然而，即使进行编目，在尝试访问 DB2 UDB 版本 7 服务器时，没有任何功能能够起作用。另外，可以将 DB2 UDB 版本 7 概要文件导入到 DB2 UDB 版本 8 服务器中，或者将DB2 UDB 版本 8 概要文件导入到 DB2 UDB 版本 7 服务器中。然而，所有其他“配置助手”功能无法与 DB2 UDB 版本 7 服务器配合工作。
- 数据仓库中心
- 复制中心
- 命令编辑器（它替代了命令中心，还包括命令中心的 Web 版本）

  将脚本导入和保存至 DB2 UDB 版本 7 服务器和从 DB2 UDB 版本 7 服务器导入脚本是不可能的。任何需要 ATTACH 的实用程序都无法工作。
- SQL 助手
- Visual Explain

通常，对于 DB2 UDB 版本 7 和更早版本的服务器，仅从“控制中心”的导航树中启动的任何 DB2 UDB 版本 8 工具或基于这些工具的任何详细视图都不可用或不可访问。当使用 DB2 UDB 版本 7 或更早版本的服务器时，应考虑使用 DB2 UDB 版本 7 工具。

可滚动游标支持:

Windows	UNIX

更改:

在版本 8 中，对于从版本 8 DB2 UDB Unix 版和 Windows 版客户机至版本 7 DB2 UDB Unix 版和 Windows 版服务器，不支持可滚动游标功能。仅对于从 DB2 UDB Unix 版和 Windows 版版本 8 客户机至 DB2 UDB Unix 版和 Windows 版版本 8 服务器，或至 DB2 UDB z/OS 和 OS/390 版版本 7 服务器，才支持可滚动游标。DB2 UDB Unix 版和 Windows 版版本 7 客户机器将仍然支持至 DB2 UDB Unix 版和 Windows 版版本 8 服务器的现有可滚动游标功能。

解决方案:

将服务器升级为版本 8。

通过 DB2 Connect V8 服务器访问 V7 服务器:

Windows	UNIX

更改:

在版本 8 中，将不支持通过版本 8 服务器从 DB2 UDB Unix 版和 Windows 版客户机访问版本 7 DB2 UDB 服务器，其中该功能由 DB2 Connect 企业版版本 8 或 DB2 UDB 企业服务器版版本 8 提供。

解决方案:

将服务器升级为版本 8。

与 CLP 和嵌入式 SQL 建立类型 1 连接:

Windows	UNIX

更改:

在 DB2 的先前版本中，当正在使用的“命令行处理器”（CLP）或嵌入式 SQL，并且与数据库建立了类型 1 连接时，在工作单元期间尝试连接至另一个数据库将失败，并产生 SQL0752N 错误。在版本 8 中，落实了工作单元，复位了连接，并允许连接至第二个数据库。即使关闭了 AUTOCOMMIT，也将落实工作单元并复位连接。

消息:

返回 DB2 Connect 消息而不是 DB2 消息:

Windows	UNIX

更改:

在版本 8 中，在前发行版中应返回 DB2 消息的情况现在可能将返回 DB2 Connect 消息。

受此更改影响的消息与绑定、连接或安全性错误相关。查询和其他 SQL 请求的 SQL 错误不受此更改的影响。

示例:

- 将返回 SQLCODE -30081 而不是返回 SQLCODE -1224
- 将返回 SQLCODE -30082 而不是返回 SQLCODE -1403
- 将返回 SQLCODE -30104 而不是返回 SQLCODE -4930

症状:

编码为对 DB2 消息作出反应的应用程序的行为可能与以前不同。

配置参数:

废弃的数据库管理器配置参数:

Windows	UNIX

更改:

下列数据库管理器配置参数已废弃:

- *backbufsz*: 在先前版本中，可以使用缺省缓冲区大小执行备份操作，且 *backbufsz* 的值将被用作缺省值。在版本 8 中，当您使用 backup 实用程序时，应显式指定备份缓冲区的大小。
- *dft_client_adpt*: 不再支持 DCE 目录服务
- *dft_client_comm*: 不再支持 DCE 目录服务
- *dir_obj_name*: 不再支持 DCE 目录服务
- *dir_path_name*: 不再支持 DCE 目录服务
- *dir_type*: 不再支持 DCE 目录服务
- *dos_rqrioblk*
- *drda_heap_sz*
- *fcm_num_anchors*、*fcm_num_connect* 和 *fcm_num_rqb*: DB2 现在将动态并自动调整消息锚点、连接条目和请求块，所以您不必调整这些参数
- *fileserver*: 不再支持 IPX/SPX
- *initdari_jvm*: Java 存储过程现在在缺省情况下将以多线程方式运行，并在不同于其他语言例程的进程中运行，因此此参数不再受支持
- *ipx_socket*: 不再支持 IPX/SPX
- *jdk11_path*: 已被 *jdk_path* 数据库管理器配置参数替换
- *keepdari*: 已被 *keepfenced* 数据库管理器配置参数替换
- *max_logicagents*: 已被 *max_connections* 数据库管理器配置参数替换
- *maxdari*: 已被 *fenced_pool* 数据库管理器配置参数替换
- *num_initdaris*: 已被 *num_initfenced* 数据库管理器配置参数替换
- *objectname*: 不再支持 IPX/SPX

- *restbufsz*: 在先前版本中，可以使用缺省缓冲区大小执行复原操作，并且可将 *restbufsz* 的值用作缺省值。在版本 8 中，当使用 restore 实用程序时，应显式指定复原缓冲区的大小。
- *route_obj_name*: 不再支持 DCE 目录服务
- *ss_logon*: 这是 OS/2® 参数，而 OS/2 不再受支持
- *udf_mem_sz*: UDF 不再在共享内存中传送数据，所以不再支持此参数

解决方案:

从应用程序中除去对这些参数的所有引用。

废弃的数据库配置参数:

Windows	UNIX

更改:

下列数据库配置参数已过时:

- *buffpage*: 在先前版本中，可以使用缺省大小创建或改变缓冲池，并且将 *buffpage* 的值用作缺省值。在版本 8 中，应在 ALTER BUFFERPOOL 或 CREATE BUFFERPOOL 语句中使用 SIZE 关键字显式指定缓冲池的大小。
- *copyprotect*
- *indexsort*

解决方案:

从应用程序中除去对这些参数的所有引用。

相关概念:

- 第 242 页的『版本 9 与前发行版和已更改的行为不兼容』

相关参考:

- 第 227 页的『不推荐使用和不继续使用的功能』

附录 B. 本地语言支持（NLS）

本节包含有关 DB2 数据库提供的本地语言支持（NLS）的信息（包括有关地域、语言和受支持的代码页（代码集）的信息），并介绍如何在数据库和应用程序中配置和使用 DB2 NLS 功能部件。

本地语言版本

DB2 数据库 Linux 版、UNIX 版和 Windows 版版本 9.1 有简体中文、繁体中文、捷克语、丹麦语、英语、芬兰语、法语、德语、意大利语、日语、韩国语、挪威语、波兰语、巴西葡萄牙语、俄语、西班牙语和瑞典语版本。

DB2 运行时客户机有下列其他语言版本：阿拉伯语、保加利亚语、克罗地亚语、荷兰语、希腊语、希伯莱语、匈牙利语、葡萄牙语、罗马尼亚语、斯洛伐克语、斯洛文尼亚语和土耳其语。

相关参考:

• 第 291 页的『受支持的地域代码和代码页』

受支持的地域代码和代码页

下列各表说明了受数据库服务器支持的语言和代码集，并说明了如何将这些值映射至数据库管理器使用的地域代码和代码页值。

注: 创建数据库时，可以使用在任何受支持平台上受支持的任何代码页。

下面是对这些表中各列的解释:

- **代码页**显示 IBM 定义的代码页，它是从操作系统代码集映射而来的。
- **组**显示代码页是单字节（“S”）、双字节（“D”）还是中性（“N”）。“-n”是一个用于创建字母 － 数字组合的数。匹配组合显示 DB2 数据库系统在何处允许连接和转换。例如，所有的“S-1”个组都可一起工作。但是，如果该组是中性的，则允许使用列示的任何其他代码页进行连接和转换。
- **代码集**显示与受支持的语言相关的代码集。代码集被映射为 DB2 代码页。
- **地域代码**显示数据库管理器为提供特定于区域的支持在内部使用的代码。
- **语言环境**显示数据库管理器所支持的语言环境值。
- **操作系统**显示支持这些语言和代码集的操作系统。当在此列中使用“主机”一词时，它指的是本身支持 EBCDIC 代码页的操作系统，例如，z/OS。注意，Linux on z/OS 不是主机平台。不能使用 DB2 数据库管理器来创建使用主机代码页的数据库，但可以使用 DB2 数据库管理器来连接至使用受支持主机代码页的主机数据库。

表 42. Unicode

代码页	组	代码集	地域代码	语言环境	操作系统
1200	N-1	16 位 Unicode	任意	任意	任意

表 42. Unicode（续）

代码页	组	代码集	地域代码	语言环境	操作系统
1208	N-1	UTF-8 编码的 Unicode	任意	任意	任意

表 43. 阿尔巴尼亚，地域标识：AL

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	355	sq_AL	AIX
850	S-1	IBM-850	355	-	AIX
923	S-1	ISO8859-15	355	sq_AL.8859-15	AIX
1208	N-1	UTF-8	355	SQ_AL	AIX
37	S-1	IBM-37	355	-	主机
1140	S-1	IBM-1140	355	-	主机
819	S-1	iso88591	355	-	HP-UX
923	S-1	iso885915	355	-	HP-UX
1051	S-1	roman8	355	-	HP-UX
437	S-1	IBM-437	355	-	OS/2
850	S-1	IBM-850	355	-	OS/2
819	S-1	ISO8859-1	355	-	Solaris
923	S-1	ISO8859-15	355	-	Solaris
1252	S-1	1252	355	-	Windows

表 44. 阿拉伯国家或地区，地域标识： AA

代码页	组	代码集	地域代码	语言环境	操作系统
1046	S-6	IBM-1046	785	Ar_AA	AIX
1089	S-6	ISO8859-6	785	ar_AA	AIX
1208	N-1	UTF-8	785	AR_AA	AIX
420	S-6	IBM-420	785	-	主机
425	S-6	IBM-425	785	-	主机
1089	S-6	iso88596	785	ar_SA.iso88596	HP-UX
864	S-6	IBM-864	785	-	OS/2
1256	S-6	1256	785	-	Windows

表 45. 澳大利亚，地域标识：AU

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	61	en_AU	AIX
850	S-1	IBM-850	61	-	AIX
923	S-1	ISO8859-15	61	en_AU.8859-15	AIX
1208	N-1	UTF-8	61	EN_AU	AIX
37	S-1	IBM-37	61	-	主机
1140	S-1	IBM-1140	61	-	主机
819	S-1	iso88591	61	-	HP-UX
923	S-1	iso885915	61	-	HP-UX
1051	S-1	roman8	61	-	HP-UX
437	S-1	IBM-437	61	-	OS/2
850	S-1	IBM-850	61	-	OS/2
819	S-1	ISO8859-1	61	en_AU	SCO
819	S-1	ISO8859-1	61	en_AU	Solaris
923	S-1	ISO8859-15	61	-	Solaris
1252	S-1	1252	61	-	Windows

表 46. 澳地利，地域标识：AT

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	43	-	AIX
850	S-1	IBM-850	43	-	AIX
923	S-1	ISO8859-15	43	-	AIX
1208	N-1	UTF-8	43	-	AIX
37	S-1	IBM-37	43	-	主机
1140	S-1	IBM-1140	43	-	主机
819	S-1	iso88591	43	-	HP-UX
923	S-1	iso885915	43	-	HP-UX
1051	S-1	roman8	43	-	HP-UX
819	S-1	ISO-8859-1	43	de_AT	Linux
923	S-1	ISO-8859-15	43	de_AT@euro	Linux
437	S-1	IBM-437	43	-	OS/2
850	S-1	IBM-850	43	-	OS/2
819	S-1	ISO8859-1	43	de_AT	SCO
819	S-1	ISO8859-1	43	de_AT	Solaris
923	S-1	ISO8859-15	43	de_AT.ISO8859-15	Solaris
1252	S-1	1252	43	-	Windows

表 47. 白俄罗斯，地域标识：BY

代码页	组	代码集	地域代码	语言环境	操作系统
1167	S-5	KOI8-RU	375	–	–
915	S-5	ISO8859-5	375	be_BY	AIX
1208	N-1	UTF-8	375	BE_BY	AIX
1025	S-5	IBM-1025	375	-	主机
1154	S-5	IBM-1154	375	-	主机
915	S-5	ISO8859-5	375	-	OS/2
1131	S-5	IBM-1131	375	-	OS/2
1251	S-5	1251	375	-	Windows

表 48. 比利时，地域标识：BE

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	32	fr_BE	AIX
819	S-1	ISO8859-1	32	nl_BE	AIX
850	S-1	IBM-850	32	Fr_BE	AIX
850	S-1	IBM-850	32	Nl_BE	AIX
923	S-1	ISO8859-15	32	fr_BE.8859-15	AIX
923	S-1	ISO8859-15	32	nl_BE.8859-15	AIX
1208	N-1	UTF-8	32	FR_BE	AIX
1208	N-1	UTF-8	32	NL_BE	AIX
274	S-1	IBM-274	32	-	主机
500	S-1	IBM-500	32	-	主机
1148	S-1	IBM-1148	32	-	主机
819	S-1	iso88591	32	-	HP-UX
923	S-1	iso885915	32	-	HP-UX
819	S-1	ISO-8859-1	32	fr_BE	Linux
819	S-1	ISO-8859-1	32	nl_BE	Linux
923	S-1	ISO-8859-15	32	fr_BE@euro	Linux
923	S-1	ISO-8859-15	32	nl_BE@euro	Linux
437	S-1	IBM-437	32	-	OS/2
850	S-1	IBM-850	32	-	OS/2

表 48. 比利时，地域标识：BE（续）

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	32	fr_BE	SCO
819	S-1	ISO8859-1	32	nl_BE	SCO
819	S-1	ISO8859-1	32	fr_BE	Solaris
819	S-1	ISO8859-1	32	nl_BE	Solaris
923	S-1	ISO8859-15	32	fr_BE.ISO8859-15	Solaris
923	S-1	ISO8859-15	32	nl_BE.ISO8859-15	Solaris
1252	S-1	1252	32	-	Windows

表 49. 保加利亚，地域标识：BG

代码页	组	代码集	地域代码	语言环境	操作系统
915	S-5	ISO8859-5	359	bg_BG	AIX
1208	N-1	UTF-8	359	BG_BG	AIX
1025	S-5	IBM-1025	359	-	主机
1154	S-5	IBM-1154	359	-	主机
915	S-5	iso88595	359	bg_BG.iso88595	HP-UX
855	S-5	IBM-855	359	-	OS/2
915	S-5	ISO8859-5	359	-	OS/2
1251	S-5	1251	359	-	Windows

表 50. 巴西，地域标识：BR

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	55	pt_BR	AIX
850	S-1	IBM-850	55	-	AIX
923	S-1	ISO8859-15	55	pt_BR.8859-15	AIX
1208	N-1	UTF-8	55	PT_BR	AIX
37	S-1	IBM-37	55	-	主机
1140	S-1	IBM-1140	55	-	主机
819	S-1	ISO8859-1	55	-	HP-UX
923	S-1	ISO8859-15	55	-	HP-UX
819	S-1	ISO-8859-1	55	pt_BR	Linux
923	S-1	ISO-8859-15	55	-	Linux
850	S-1	IBM-850	55	-	OS/2
819	S-1	ISO8859-1	55	pt_BR	SCO
819	S-1	ISO8859-1	55	pt_BR	Solaris
923	S-1	ISO8859-15	55	-	Solaris
1252	S-1	1252	55	-	Windows

表 51. 加拿大，地域标识：CA

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	1	fr_CA	AIX
850	S-1	IBM-850	1	Fr_CA	AIX
923	S-1	ISO8859-15	1	fr_CA.8859-15	AIX
1208	N-1	UTF-8	1	FR_CA	AIX
37	S-1	IBM-37	1	-	主机
1140	S-1	IBM-1140	1	-	主机
819	S-1	iso88591	1	fr_CA.iso88591	HP-UX
923	S-1	iso885915	1	-	HP-UX
1051	S-1	roman8	1	fr_CA.roman8	HP-UX
819	S-1	ISO-8859-1	1	en_CA	Linux

表 51. 加拿大，地域标识：CA（续）

代码页	组	代码集	地域代码	语言环境	操作系统
923	S-1	ISO-8859-15	1	-	Linux
850	S-1	IBM-850	1	-	OS/2
819	S-1	ISO8859-1	1	en_CA	SCO
819	S-1	ISO8859-1	1	fr_CA	SCO
819	S-1	ISO8859-1	1	en_CA	Solaris
923	S-1	ISO8859-15	1	-	Solaris
1252	S-1	1252	1	-	Windows

表 52. 加拿大（法语区），地域标识：CA

代码页	组	代码集	地域代码	语言环境	操作系统
863	S-1	IBM-863	2	-	OS/2

表 53. 中华人民共和国（PRC），地域标识：CN

代码页	组	代码集	地域代码	语言环境	操作系统
1383	D-4	IBM-eucCN	86	zh_CN	AIX
1386	D-4	GBK	86	Zh_CN.GBK	AIX
1208	N-1	UTF-8	86	ZH_CN	AIX
935	D-4	IBM-935	86	-	主机
1388	D-4	IBM-1388	86	-	主机
1383	D-4	hp15CN	86	zh_CN.hp15CN	HP-UX
1386	D-4	GBK	86	zh_CN.GBK	Linux
1381	D-4	IBM-1381	86	-	OS/2
1386	D-4	GBK	86	-	OS/2
1383	D-4	eucCN	86	zh_CN	SCO
1383	D-4	eucCN	86	zh_CN.eucCN	SCO
1383	D-4	gb2312	86	zh	Solaris
1208	N-1	UTF-8	86	zh.UTF-8	Solaris
1381	D-4	IBM-1381	86	-	Windows
1386	D-4	GBK	86	-	Windows
1392/5488	D-4		86	-	
请参阅“注”（第 309 页的 1）。					

表 54. 克罗地亚，地域标识：HR

代码页	组	代码集	地域代码	语言环境	操作系统
912	S-2	ISO8859-2	385	hr_HR	AIX
1208	N-1	UTF-8	385	HR_HR	AIX
870	S-2	IBM-870	385	-	主机
1153	S-2	IBM-1153	385	-	主机
912	S-2	iso88592	385	hr_HR.iso88592	HP-UX
912	S-2	ISO-8859-2	385	hr_HR	Linux
852	S-2	IBM-852	385	-	OS/2
912	S-2	ISO8859-2	385	hr_HR.ISO8859-2	SCO
1250	S-2	1250	385	-	Windows

表 55. 捷克共和国，地域标识：CZ

代码页	组	代码集	地域代码	语言环境	操作系统
912	S-2	ISO8859-2	421	cs_CZ	AIX
1208	N-1	UTF-8	421	CS_CZ	AIX

表 55. 捷克共和国，地域标识：CZ (续)

代码页	组	代码集	地域代码	语言环境	操作系统
870	S-2	IBM-870	421	-	主机
1153	S-2	IBM-1153	421	-	主机
912	S-2	iso88592	421	cs_CZ.iso88592	HP-UX
912	S-2	ISO-8859-2	421	cs_CZ	Linux
852	S-2	IBM-852	421	-	OS/2
912	S-2	ISO8859-2	421	cs_CZ.ISO8859-2	SCO
1250	S-2	1250	421	-	Windows

表 56. 丹麦，地域标识：DK

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	45	da_DK	AIX
850	S-1	IBM-850	45	Da_DK	AIX
923	S-1	ISO8859-15	45	da_DK.8859-15	AIX
1208	N-1	UTF-8	45	DA_DK	AIX
277	S-1	IBM-277	45	-	主机
1142	S-1	IBM-1142	45	-	主机
819	S-1	iso88591	45	da_DK.iso88591	HP-UX
923	S-1	iso885915	45	_	HP-UX
1051	S-1	roman8	45	da_DK.roman8	HP-UX
819	S-1	ISO-8859-1	45	da_DK	Linux
923	S-1	ISO-8859-15	45	-	Linux
850	S-1	IBM-850	45	-	OS/2
819	S-1	ISO8859-1	45	da	SCO
819	S-1	ISO8859-1	45	da_DA	SCO
819	S-1	ISO8859-1	45	da_DK	SCO
819	S-1	ISO8859-1	45	da	Solaris
923	S-1	ISO8859-15	45	da.ISO8859-15	Solaris
1252	S-1	1252	45	-	Windows

表 57. 爱沙尼亚，地域标识：EE

代码页	组	代码集	地域代码	语言环境	操作系统
922	S-10	IBM-922	372	Et_EE	AIX
1208	N-1	UTF-8	372	ET_EE	AIX
1122	S-10	IBM-1122	372	-	主机
1157	S-10	IBM-1157	372	-	主机
922	S-10	IBM-922	372	-	OS/2
1257	S-10	1257	372	-	Windows

表 58. 芬兰，地域标识：FI

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	358	fi_FI	AIX
850	S-1	IBM-850	358	Fi_FI	AIX
923	S-1	ISO8859-15	358	fi_FI.8859-15	AIX
1208	N-1	UTF-8	358	FI_FI	AIX
278	S-1	IBM-278	358	-	主机
1143	S-1	IBM-1143	358	-	主机
819	S-1	iso88591	358	fi_FI.iso88591	HP-UX
923	S-1	iso885915	358	-	HP-UX
1051	S-1	roman8	358	fi-FI.roman8	HP-UX

表 58. 芬兰，地域标识：FI (续)

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO-8859-1	358	fi_FI	Linux
923	S-1	ISO-8859-15	358	fi_FI@euro	Linux
437	S-1	IBM-437	358	-	OS/2
850	S-1	IBM-850	358	-	OS/2
819	S-1	ISO8859-1	358		SCO
819	S-1	ISO8859-1	358	fi_FI	SCO
819	S-1	ISO8859-1	358	sv_FI	SCO
819	S-1	ISO8859-1	358	-	Solaris
923	S-1	ISO8859-15	358	fi.ISO8859-15	Solaris
1252	S-1	1252	358	-	Windows

表 59. FYR 马其顿，地域标识：MK

代码页	组	代码集	地域代码	语言环境	操作系统
915	S-5	ISO8859-5	389	mk_MK	AIX
1208	N-1	UTF-8	389	MK_MK	AIX
1025	S-5	IBM-1025	389	-	主机
1154	S-5	IBM-1154	389	-	主机
915	S-5	iso88595	389	-	HP-UX
855	S-5	IBM-855	389	-	OS/2
915	S-5	ISO8859-5	389	-	OS/2
1251	S-5	1251	389	-	Windows

表 60. 法国，地域标识：FR

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	33	fr_FR	AIX
850	S-1	IBM-850	33	Fr_FR	AIX
923	S-1	ISO8859-15	33	fr_FR.8859-15	AIX
1208	N-1	UTF-8	33	FR_FR	AIX
297	S-1	IBM-297	33	-	主机
1147	S-1	IBM-1147	33	-	主机
819	S-1	iso88591	33	fr_FR.iso88591	HP-UX
923	S-1	iso885915	33	-	HP-UX
1051	S-1	roman8	33	fr_FR.roman8	HP-UX
819	S-1	ISO-8859-1	33	fr_FR	Linux
923	S-1	ISO-8859-15	33	fr_FR@euro	Linux
437	S-1	IBM-437	33	-	OS/2
850	S-1	IBM-850	33	-	OS/2
819	S-1	ISO8859-1	33		SCO
819	S-1	ISO8859-1	33	fr_FR	SCO
819	S-1	ISO8859-1	33		Solaris
923	S-1	ISO8859-15	33	fr.ISO8859-15	Solaris
1208	N-1	UTF-8	33	fr.UTF-8	Solaris
1252	S-1	1252	33	-	Windows

表 61. 德国，地域标识：DE

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	49	de_DE	AIX
850	S-1	IBM-850	49	De_DE	AIX
923	S-1	ISO8859-15	49	de_DE.8859-15	AIX

表 61. 德国，地域标识：DE（续）

代码页	组	代码集	地域代码	语言环境	操作系统
1208	N-1	UTF-8	49	DE_DE	AIX
273	S-1	IBM-273	49	-	主机
1141	S-1	IBM-1141	49	-	主机
819	S-1	iso88591	49	de_DE.iso88591	HP-UX
923	S-1	iso885915	49	_	HP-UX
1051	S-1	roman8	49	de_DE.roman8	HP-UX
819	S-1	ISO-8859-1	49	de_DE	Linux
923	S-1	ISO-8859-15	49	de_DE@euro	Linux
437	S-1	IBM-437	49	-	OS/2
850	S-1	IBM-850	49	-	OS/2
819	S-1	ISO8859-1	49		SCO
819	S-1	ISO8859-1	49	de_DE	SCO
819	S-1	ISO8859-1	49		Solaris
923	S-1	ISO8859-15	49	de.ISO8859-15	Solaris
1208	N-1	UTF-8	49	de.UTF-8	Solaris
1252	S-1	1252	49	-	Windows

表 62. 希腊，地域标识：GR

代码页	组	代码集	地域代码	语言环境	操作系统
813	S-7	ISO8859-7	30	el_GR	AIX
1208	N-1	UTF-8	30	EL_GR	AIX
423	S-7	IBM-423	30	-	主机
875	S-7	IBM-875	30	-	主机
813	S-7	iso88597	30	el_GR.iso88597	HP-UX
813	S-7	ISO-8859-7	30	el_GR	Linux
813	S-7	ISO8859-7	30	-	OS/2
869	S-7	IBM-869	30	-	OS/2
813	S-7	ISO8859-7	30	el_GR.ISO8859-7	SCO
737	S-7	737	30	-	Windows
1253	S-7	1253	30	-	Windows

表 63. 匈牙利，地域标识：HU

代码页	组	代码集	地域代码	语言环境	操作系统
912	S-2	ISO8859-2	36	hu_HU	AIX
1208	N-1	UTF-8	36	HU_HU	AIX
870	S-2	IBM-870	36	-	主机
1153	S-2	IBM-1153	36	-	主机
912	S-2	iso88592	36	hu_HU.iso88592	HP-UX
912	S-2	ISO-8859-2	36	hu_HU	Linux
852	S-2	IBM-852	36	-	OS/2
912	S-2	ISO8859-2	36	hu_HU.ISO8859-2	SCO
1250	S-2	1250	36	-	Windows

表 64. 冰岛，地域标识：IS

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	354	is_IS	AIX
850	S-1	IBM-850	354	Is_IS	AIX
923	S-1	ISO8859-15	354	is_IS.8859-15	AIX
1208	N-1	UTF-8	354	IS_IS	AIX

表 64. 冰岛，地域标识：IS（续）

代码页	组	代码集	地域代码	语言环境	操作系统
871	S-1	IBM-871	354	-	主机
1149	S-1	IBM-1149	354	-	主机
819	S-1	iso88591	354	is_IS.iso88591	HP-UX
923	S-1	iso885915	354	-	HP-UX
1051	S-1	roman8	354	is_IS.roman8	HP-UX
819	S-1	ISO-8859-1	354	is_IS	Linux
923	S-1	ISO-8859-15	354	-	Linux
850	S-1	IBM-850	354	-	OS/2
819	S-1	ISO8859-1	354		SCO
819	S-1	ISO8859-1	354	is_IS	SCO
819	S-1	ISO8859-1	354	-	Solaris
923	S-1	ISO8859-15	354	-	Solaris
1252	S-1	1252	354	-	Windows

表 65. 印度，地域标识：IN

代码页	组	代码集	地域代码	语言环境	操作系统
806	S-13	IBM-806	91	hi_IN	-
1137	S-13	IBM-1137	91	-	主机

表 66. 印度尼西亚，地域标识：ID

代码页	组	代码集	地域代码	语言环境	操作系统
1252	S-1	1252	62	-	Windows

表 67. 爱尔兰，地域标识：IE

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	353	-	AIX
850	S-1	IBM-850	353	-	AIX
923	S-1	ISO8859-15	353	-	AIX
1208	N-1	UTF-8	353	-	AIX
285	S-1	IBM-285	353	-	主机
1146	S-1	IBM-1146	353	-	主机
819	S-1	iso88591	353	-	HP-UX
923	S-1	iso885915	353	-	HP-UX
1051	S-1	roman8	353	-	HP-UX
819	S-1	ISO-8859-1	353	en_IE	Linux
923	S-1	ISO-8859-15	353	en_IE@euro	Linux
437	S-1	IBM-437	353	-	OS/2
850	S-1	IBM-850	353	-	OS/2
819	S-1	ISO8859-1	353	en_IE.ISO8859-1	SCO
819	S-1	ISO8859-1	353	en_IE	Solaris
923	S-1	ISO8859-15	353	en_IE.ISO8859-15	Solaris
1252	S-1	1252	353	-	Windows

表 68. 以色列，地域标识：IL

代码页	组	代码集	地域代码	语言环境	操作系统
856	S-8	IBM-856	972	Iw_IL	AIX
916	S-8	ISO8859-8	972	iw_IL	AIX
1208	N-1	UTF-8	972	HE-IL	AIX

表 68. 以色列，地域标识：IL（续）

代码页	组	代码集	地域代码	语言环境	操作系统
916	S-8	ISO-8859-8	972	iw_IL	Linux
424	S-8	IBM-424	972	-	主机
862	S-8	IBM-862	972	-	OS/2
1255	S-8	1255	972	-	Windows

表 69. 意大利，地域标识：IT

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	39	it_IT	AIX
850	S-1	IBM-850	39	It_IT	AIX
923	S-1	ISO8859-15	39	it_IT.8859-15	AIX
1208	N-1	UTF-8	39	It_IT	AIX
280	S-1	IBM-280	39	-	主机
1144	S-1	IBM-1144	39	-	主机
819	S-1	iso88591	39	it_IT.iso88591	HP-UX
923	S-1	iso885915	39	_	HP-UX
1051	S-1	roman8	39	it_IT.roman8	HP-UX
819	S-1	ISO-8859-1	39	it_IT	Linux
923	S-1	ISO-8859-15	39	it_IT@euro	Linux
437	S-1	IBM-437	39	-	OS/2
850	S-1	IBM-850	39	-	OS/2
819	S-1	ISO8859-1	39		SCO
819	S-1	ISO8859-1	39	it_IT	SCO
819	S-1	ISO8859-1	39		Solaris
923	S-1	ISO8859-15	39	it.ISO8859-15	Solaris
1208	N-1	UTF-8	39	it.UTF-8	Solaris
1252	S-1	1252	39	-	Windows

表 70. 日本，地域标识：JP

代码页	组	代码集	地域代码	语言环境	操作系统
932	D-1	IBM-932	81	Ja_JP	AIX
943	D-1	IBM-943	81	Ja_JP	AIX
请参阅“注”（第 309 页的 2）。					
954	D-1	IBM-eucJP	81	ja_JP	AIX
1208	N-1	UTF-8	81	JA_JP	AIX
930	D-1	IBM-930	81	-	主机
939	D-1	IBM-939	81	-	主机
5026	D-1	IBM-5026	81	-	主机
5035	D-1	IBM-5035	81	-	主机
1390	D-1		81	-	主机
1399	D-1		81	-	主机
954	D-1	eucJP	81	ja_JP.eucJP	HP-UX
5039	D-1	SJIS	81	ja_JP.SJIS	HP-UX
954	D-1	EUC-JP	81	ja_JP	Linux
932	D-1	IBM-932	81	-	OS/2
942	D-1	IBM-942	81	-	OS/2
943	D-1	IBM-943	81	-	OS/2
954	D-1	eucJP	81	ja	SCO
954	D-1	eucJP	81	ja_JP	SCO
954	D-1	eucJP	81	ja_JP.EUC	SCO
954	D-1	eucJP	81	ja_JP.eucJP	SCO

表 70. 日本，地域标识：JP（续）

代码页	组	代码集	地域代码	语言环境	操作系统
943	D-1	IBM-943	81	ja_JP.PCK	Solaris
954	D-1	eucJP	81	ja	Solaris
1208	N-1	UTF-8	81	ja_JP.UTF-8	Solaris
943	D-1	IBM-943	81	-	Windows
1394	D-1		81	-	

请参阅“注”（第 309 页的 3）。

表 71. 哈萨克斯坦，地域标识：KZ

代码页	组	代码集	地域代码	语言环境	操作系统
1251	S-5	1251	7	-	Windows

表 72. 韩国，地域标识：KR

代码页	组	代码集	地域代码	语言环境	操作系统
970	D-3	IBM-eucKR	82	ko_KR	AIX
1208	N-1	UTF-8	82	KO_KR	AIX
933	D-3	IBM-933	82	-	主机
1364	D-3	IBM-1364	82	-	主机
970	D-3	eucKR	82	ko_KR.eucKR	HP-UX
970	D-3	EUC-KR	82	ko_KR	Linux
949	D-3	IBM-949	82	-	OS/2
970	D-3	eucKR	82	ko_KR.eucKR	SGI
970	D-3	5601	82	ko	Solaris
1208	N-1	UTF-8	82	ko.UTF-8	Solaris
1363	D-3	1363	82	-	Windows

表 73. 拉丁美洲，地域标识：Lat

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	3	-	AIX
850	S-1	IBM-850	3	-	AIX
923	S-1	ISO8859-15	3	-	AIX
1208	N-1	UTF-8	3	-	AIX
284	S-1	IBM-284	3	-	主机
1145	S-1	IBM-1145	3	-	主机
819	S-1	iso88591	3	-	HP-UX
923	S-1	iso885915	3	-	HP-UX
1051	S-1	roman8	3	-	HP-UX
819	S-1	ISO-8859-1	3	-	Linux
923	S-1	ISO-8859-15	3	-	Linux
437	S-1	IBM-437	3	-	OS/2
850	S-1	IBM-850	3	-	OS/2
819	S-1	ISO8859-1	3	-	Solaris
923	S-1	ISO8859-15	3	-	Solaris
1252	S-1	1252	3	-	Windows

表 74. 拉脱维亚，地域标识：LV

代码页	组	代码集	地域代码	语言环境	操作系统
921	S-10	IBM-921	371	Lv_LV	AIX
1208	N-1	UTF-8	371	LV_LV	AIX

表 74. 拉脱维亚，地域标识：LV（续）

代码页	组	代码集	地域代码	语言环境	操作系统
1112	S-10	IBM-1112	371	-	主机
1156	S-10	IBM-1156	371	-	主机
921	S-10	IBM-921	371	-	OS/2
1257	S-10	1257	371	-	Windows

表 75. 立陶宛，地域标识：LT

代码页	组	代码集	地域代码	语言环境	操作系统
921	S-10	IBM-921	370	Lt_LT	AIX
1208	N-1	UTF-8	370	LT_LT	AIX
1112	S-10	IBM-1112	370	-	主机
1156	S-10	IBM-1156	370	-	主机
921	S-10	IBM-921	370	-	OS/2
1257	S-10	1257	370	-	Windows

表 76. 马来西亚，地域标识：ID

代码页	组	代码集	地域代码	语言环境	操作系统
1252	S-1	1252	60	-	Windows

表 77. 荷兰，地域标识：NL

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	31	nl_NL	AIX
850	S-1	IBM-850	31	Nl_NL	AIX
923	S-1	ISO8859-15	31	nl_NL.8859-15	AIX
1208	N-1	UTF-8	31	NL_NL	AIX
37	S-1	IBM-37	31	-	主机
1140	S-1	IBM-1140	31	-	主机
819	S-1	iso88591	31	nl_NL.iso88591	HP-UX
923	S-1	iso885915	31	_	HP-UX
1051	S-1	roman8	31	nl_NL.roman8	HP-UX
819	S-1	ISO-8859-1	31	nl_NL	Linux
923	S-1	ISO-8859-15	31	nl_NL@euro	Linux
437	S-1	IBM-437	31	-	OS/2
850	S-1	IBM-850	31	-	OS/2
819	S-1	ISO8859-1	31	nl	SCO
819	S-1	ISO8859-1	31	nl_NL	SCO
819	S-1	ISO8859-1	31	nl	Solaris
923	S-1	ISO8859-15	31	nl.ISO8859-15	Solaris
1252	S-1	1252	31	-	Windows

表 78. 新西兰，地域标识：NZ

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	64	-	AIX
850	S-1	IBM-850	64	-	AIX
923	S-1	ISO8859-15	64	-	AIX
1208	N-1	UTF-8	64	-	AIX
37	S-1	IBM-37	64	-	主机
1140	S-1	IBM-1140	64	-	主机
819	S-1	ISO8859-1	64	-	HP-UX

表 78. 新西兰，地域标识：NZ (续)

代码页	组	代码集	地域代码	语言环境	操作系统
923	S-1	ISO8859-15	64	-	HP-UX
850	S-1	IBM-850	64	-	OS/2
819	S-1	ISO8859-1	64	en_NZ	SCO
819	S-1	ISO8859-1	64	en_NZ	Solaris
923	S-1	ISO8859-15	64	-	Solaris
1252	S-1	1252	64	-	Windows

表 79. 挪威，地域标识：NO

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	47	no_NO	AIX
850	S-1	IBM-850	47	No_NO	AIX
923	S-1	ISO8859-15	47	no_NO.8859-15	AIX
1208	N-1	UTF-8	47	NO_NO	AIX
277	S-1	IBM-277	47	-	主机
1142	S-1	IBM-1142	47	-	主机
819	S-1	iso88591	47	no_NO.iso88591	HP-UX
923	S-1	iso885915	47	-	HP-UX
1051	S-1	roman8	47	no_NO.roman8	HP-UX
819	S-1	ISO-8859-1	47	no_NO	Linux
923	S-1	ISO-8859-15	47	-	Linux
850	S-1	IBM-850	47	-	OS/2
819	S-1	ISO8859-1	47	no	SCO
819	S-1	ISO8859-1	47	no_NO	SCO
819	S-1	ISO8859-1	47	no	Solaris
923	S-1	ISO8859-15	47	-	Solaris
1252	S-1	1252	47	-	Windows

表 80. 波兰，地域标识：PL

代码页	组	代码集	地域代码	语言环境	操作系统
912	S-2	ISO8859-2	48	pl_PL	AIX
1208	N-1	UTF-8	48	PL_PL	AIX
870	S-2	IBM-870	48	-	主机
1153	S-2	IBM-1153	48	-	主机
912	S-2	iso88592	48	pl_PL.iso88592	HP-UX
912	S-2	ISO-8859-2	48	pl_PL	Linux
852	S-2	IBM-852	48	-	OS/2
912	S-2	ISO8859-2	48	pl_PL.ISO8859-2	SCO
1250	S-2	1250	48	-	Windows

表 81. 葡萄牙，地域标识：PT

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	351	pt_PT	AIX
850	S-1	IBM-850	351	Pt_PT	AIX
923	S-1	ISO8859-15	351	pt_PT.8859-15	AIX
1208	N-1	UTF-8	351	PT_PT	AIX
37	S-1	IBM-37	351	-	主机
1140	S-1	IBM-1140	351	-	主机
819	S-1	iso88591	351	pt_PT.iso88591	HP-UX
923	S-1	iso885915	351	-	HP-UX

表 81. 葡萄牙，地域标识：PT（续）

代码页	组	代码集	地域代码	语言环境	操作系统
1051	S-1	roman8	351	pt_PT.roman8	HP-UX
819	S-1	ISO-8859-1	351	pt_PT	Linux
923	S-1	ISO-8859-15	351	pt_PT@euro	Linux
850	S-1	IBM-850	351	-	OS/2
860	S-1	IBM-860	351	-	OS/2
819	S-1	ISO8859-1	351	pt	SCO
819	S-1	ISO8859-1	351	pt_PT	SCO
819	S-1	ISO8859-1	351	pt	Solaris
923	S-1	ISO8859-15	351	pt.ISO8859-15	Solaris
1252	S-1	1252	351	-	Windows

表 82. 罗马尼亚，地域标识：RO

代码页	组	代码集	地域代码	语言环境	操作系统
912	S-2	ISO8859-2	40	ro_RO	AIX
1208	N-1	UTF-8	40	RO_RO	AIX
870	S-2	IBM-870	40	-	主机
1153	S-2	IBM-1153	40	-	主机
912	S-2	iso88592	40	ro_RO.iso88592	HP-UX
912	S-2	ISO-8859-2	40	ro_RO	Linux
852	S-2	IBM-852	40	-	OS/2
912	S-2	ISO8859-2	40	ro_RO.ISO8859-2	SCO
1250	S-2	1250	40	-	Windows

表 83. 俄罗斯，地域标识：RU

代码页	组	代码集	地域代码	语言环境	操作系统
915	S-5	ISO8859-5	7	ru_RU	AIX
1208	N-1	UTF-8	7	RU_RU	AIX
1025	S-5	IBM-1025	7	-	主机
1154	S-5	IBM-1154	7	-	主机
915	S-5	iso88595	7	ru_RU.iso88595	HP-UX
878	S-5	KOI8-R	7	ru_RU.koi8-r	Linux 和 Solaris
915	S-5	ISO-8859-5	7	ru_RU	Linux
866	S-5	IBM-866	7	-	OS/2
915	S-5	ISO8859-5	7	-	OS/2
915	S-5	ISO8859-5	7	ru_RU.ISO8859-5	SCO
1251	S-5	1251	7	-	Windows

表 84. 塞尔维亚／黑山，地域标识：SP

代码页	组	代码集	地域代码	语言环境	操作系统
915	S-5	ISO8859-5	381	sr_SP	AIX
1208	N-1	UTF-8	381	SR_SP	AIX
1025	S-5	IBM-1025	381	-	主机
1154	S-5	IBM-1154	381	-	主机
915	S-5	iso88595	381	-	HP-UX
855	S-5	IBM-855	381	-	OS/2
915	S-5	ISO8859-5	381	-	OS/2
1251	S-5	1251	381	-	Windows

表 85. 斯洛伐克，地域标识：SK

代码页	组	代码集	地域代码	语言环境	操作系统
912	S-2	ISO8859-2	422	sk_SK	AIX
1208	N-1	UTF-8	422	SK_SK	AIX
870	S-2	IBM-870	422	-	主机
1153	S-2	IBM-1153	422	-	主机
912	S-2	iso88592	422	sk_SK.iso88592	HP-UX
852	S-2	IBM-852	422	-	OS/2
912	S-2	ISO8859-2	422	sk_SK.ISO8859-2	SCO
1250	S-2	1250	422	-	Windows

表 86. 斯洛文尼亚，地域标识：SI

代码页	组	代码集	地域代码	语言环境	操作系统
912	S-2	ISO8859-2	386	sl_SI	AIX
1208	N-1	UTF-8	386	SL_SI	AIX
870	S-2	IBM-870	386	-	主机
1153	S-2	IBM-1153	386	-	主机
912	S-2	iso88592	386	sl_SI.iso88592	HP-UX
912	S-2	ISO-8859-2	386	sl_SI	Linux
852	S-2	IBM-852	386	-	OS/2
912	S-2	ISO8859-2	386	sl_SI.ISO8859-2	SCO
1250	S-2	1250	386	-	Windows

表 87. 南非，地域标识：ZA

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	27	en_ZA	AIX
850	S-1	IBM-850	27	En_ZA	AIX
923	S-1	ISO8859-15	27	en_ZA.8859-15	AIX
1208	N-1	UTF-8	27	EN_ZA	AIX
285	S-1	IBM-285	27	-	主机
1146	S-1	IBM-1146	27	-	主机
819	S-1	iso88591	27	-	HP-UX
923	S-1	iso885915	27	-	HP-UX
1051	S-1	roman8	27	-	HP-UX
437	S-1	IBM-437	27	-	OS/2
850	S-1	IBM-850	27	-	OS/2
819	S-1	ISO8859-1	27	en_ZA.ISO8859-1	SCO
819	S-1	ISO8859-1	27	-	Solaris
923	S-1	ISO8859-15	27	-	Solaris
1252	S-1	1252	27	-	Windows

表 88. 西班牙，地域标识：ES

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	34	es_ES	AIX
850	S-1	IBM-850	34	Es_ES	AIX
923	S-1	ISO8859-15	34	es_ES.8859-15	AIX
1208	N-1	UTF-8	34	ES_ES	AIX
284	S-1	IBM-284	34	-	主机
1145	S-1	IBM-1145	34	-	主机
819	S-1	iso88591	34	es_ES.iso88591	HP-UX
923	S-1	iso885915	34	-	HP-UX

表 88. 西班牙，地域标识：ES（续）

代码页	组	代码集	地域代码	语言环境	操作系统
1051	S-1	roman8	34	es_ES.roman8	HP-UX
819	S-1	ISO-8859-1	34	es_ES	Linux
923	S-1	ISO-8859-15	34	es_ES@euro	Linux
437	S-1	IBM-437	34	-	OS/2
850	S-1	IBM-850	34	-	OS/2
819	S-1	ISO8859-1	34	es	SCO
819	S-1	ISO8859-1	34	es_ES	SCO
819	S-1	ISO8859-1	34	es	Solaris
923	S-1	ISO8859-15	34	es.ISO8859-15	Solaris
1208	N-1	UTF-8	34	es.UTF-8	Solaris
1252	S-1	1252	34	-	Windows

表 89. 西班牙（加泰罗尼亚），地域标识：ES

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	34	ca_ES	AIX
850	S-1	IBM-850	34	Ca_ES	AIX
923	S-1	ISO8859-15	34	ca_ES.8859-15	AIX
1208	N-1	UTF-8	34	CA_ES	AIX

表 90. 瑞典，地域标识：SE

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	46	sv_SE	AIX
850	S-1	IBM-850	46	Sv_SE	AIX
923	S-1	ISO8859-15	46	sv_SE.8859-15	AIX
1208	N-1	UTF-8	46	SV_SE	AIX
278	S-1	IBM-278	46	-	主机
1143	S-1	IBM-1143	46	-	主机
819	S-1	iso88591	46	sv_SE.iso88591	HP-UX
923	S-1	iso885915	46	-	HP-UX
1051	S-1	roman8	46	sv_SE.roman8	HP-UX
819	S-1	ISO-8859-1	46	sv_SE	Linux
923	S-1	ISO-8859-15	46	-	Linux
437	S-1	IBM-437	46	-	OS/2
850	S-1	IBM-850	46	-	OS/2
819	S-1	ISO8859-1	46	sv	SCO
819	S-1	ISO8859-1	46	sv_SE	SCO
819	S-1	ISO8859-1	46	sv	Solaris
923	S-1	ISO8859-15	46	sv.ISO8859-15	Solaris
1208	N-1	UTF-8	46	sv.UTF-8	Solaris
1252	S-1	1252	46	-	Windows

表 91. 瑞士，地域标识：CH

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	41	de_CH	AIX
850	S-1	IBM-850	41	De_CH	AIX
923	S-1	ISO8859-15	41	de_CH.8859-15	AIX
1208	N-1	UTF-8	41	DE_CH	AIX
500	S-1	IBM-500	41	-	主机
1148	S-1	IBM-1148	41	-	主机

表 91. 瑞士，地域标识：CH（续）

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	iso88591	41	-	HP-UX
923	S-1	iso885915	41	-	HP-UX
1051	S-1	roman8	41	-	HP-UX
819	S-1	ISO-8859-1	41	de_CH	Linux
923	S-1	ISO-8859-15	41	-	Linux
437	S-1	IBM-437	41	-	OS/2
850	S-1	IBM-850	41	-	OS/2
819	S-1	ISO8859-1	41	de_CH	SCO
819	S-1	ISO8859-1	41	fr_CH	SCO
819	S-1	ISO8859-1	41	it_CH	SCO
819	S-1	ISO8859-1	41	de_CH	Solaris
923	S-1	ISO8859-15	41	-	Solaris
1252	S-1	1252	41	-	Windows

表 92. 台湾，地域标识：TW

代码页	组	代码集	地域代码	语言环境	操作系统
950	D-2	big5	88	Zh_TW	AIX
请参阅“注”（第 310 页的 8）。					
964	D-2	IBM-eucTW	88	zh_TW	AIX
1208	N-1	UTF-8	88	ZH_TW	AIX
937	D-2	IBM-937	88	-	主机
1371	D-2	IBM-1371	88	-	主机
950	D-2	big5	88	zh_TW.big5	HP-UX
964	D-2	eucTW	88	zh_TW.eucTW	HP-UX
950	D-2	BIG5	88	zh_TW	Linux
938	D-2	IBM-938	88	-	OS/2
948	D-2	IBM-948	88	-	OS/2
950	D-2	big5	88	-	OS/2
950	D-2	big5	88	zh_TW.BIG5	Solaris
964	D-2	cns11643	88	zh_TW	Solaris
1208	N-1	UTF-8	88	zh_TW.UTF-8	Solaris
950	D-2	big5	88	-	Windows
请参阅“注”（第 310 页的 8）。					

表 93. 泰国，地域标识：TH

代码页	组	代码集	地域代码	语言环境	操作系统
874	S-20	TIS620-1	66	th_TH	AIX
1208	N-1	UTF-8	66	TH_TH	AIX
838	S-20	IBM-838	66	-	主机
1160	S-20	IBM-1160	66	-	主机
874	S-20	tis620	66	th_TH.tis620	HP-UX
874	S-20	TIS620-1	66	-	OS/2
874	S-20	TIS620-1	66	-	Windows

表 94. 土耳其，地域标识：TR

代码页	组	代码集	地域代码	语言环境	操作系统
920	S-9	ISO8859-9	90	tr_TR	AIX
1208	N-1	UTF-8	90	TR_TR	AIX
1026	S-9	IBM-1026	90	-	主机

表 94. 土耳其，地域标识：TR（续）

代码页	组	代码集	地域代码	语言环境	操作系统
1155	S-9	IBM-1155	90	-	主机
920	S-9	iso88599	90	tr_TR.iso88599	HP-UX
920	S-9	ISO-8859-9	90	tr_TR	Linux
857	S-9	IBM-857	90	-	OS/2
920	S-9	ISO8859-9	90	tr_TR.ISO8859-9	SCO
1254	S-9	1254	90	-	Windows

表 95. 联合王国，地域标识：GB

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	44	en_GB	AIX
850	S-1	IBM-850	44	En_GB	AIX
923	S-1	ISO8859-15	44	en_GB.8859-15	AIX
1208	N-1	UTF-8	44	EN_GB	AIX
285	S-1	IBM-285	44	-	主机
1146	S-1	IBM-1146	44	-	主机
819	S-1	iso88591	44	en_GB.iso88591	HP-UX
923	S-1	iso885915	44	-	HP-UX
1051	S-1	roman8	44	en_GB.roman8	HP-UX
819	S-1	ISO-8859-1	44	en_GB	Linux
923	S-1	ISO-8859-15	44	-	Linux
437	S-1	IBM-437	44	-	OS/2
850	S-1	IBM-850	44	-	OS/2
819	S-1	ISO8859-1	44	en_GB	SCO
819	S-1	ISO8859-1	44	en	SCO
819	S-1	ISO8859-1	44	en_GB	Solaris
923	S-1	ISO8859-15	44	en_GB.ISO8859-15	Solaris
1252	S-1	1252	44	-	Windows

表 96. 乌克兰，地域标识：UA

代码页	组	代码集	地域代码	语言环境	操作系统
1124	S-12	IBM-1124	380	Uk_UA	AIX
1208	N-1	UTF-8	380	UK_UA	AIX
1123	S-12	IBM-1123	380	-	主机
1158	S-12	IBM-1158	380	-	主机
1168	S-12	KOI8-U	380	uk_UA.koi8u	Linux
1125	S-12	IBM-1125	380	-	OS/2
1251	S-12	1251	380	-	Windows

表 97. 美国，地域标识：US

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO8859-1	1	en_US	AIX
850	S-1	IBM-850	1	En_US	AIX
923	S-1	ISO8859-15	1	en_US.8859-15	AIX
1208	N-1	UTF-8	1	EN_US	AIX
37	S-1	IBM-37	1	-	主机
1140	S-1	IBM-1140	1	-	主机
819	S-1	iso88591	1	en_US.iso88591	HP-UX
923	S-1	iso885915	1	-	HP-UX
1051	S-1	roman8	1	en_US.roman8	HP-UX

表 97. 美国，地域标识：US（续）

代码页	组	代码集	地域代码	语言环境	操作系统
819	S-1	ISO-8859-1	1	en_US	Linux
923	S-1	ISO-8859-15	1	-	Linux
437	S-1	IBM-437	1	-	OS/2
850	S-1	IBM-850	1	-	OS/2
819	S-1	ISO8859-1	1	en_US	SCO
819	S-1	ISO8859-1	1	en_US	SGI
819	S-1	ISO8859-1	1	en_US	Solaris
923	S-1	ISO8859-15	1	en_US.ISO8859-15	Solaris
1208	N-1	UTF-8	1	en_US.UTF-8	Solaris
1252	S-1	1252	1	-	Windows

表 98. 越南，地域标识：VN

代码页	组	代码集	地域代码	语言环境	操作系统
1129	S-11	IBM-1129	84	Vi_VN	AIX
1208	N-1	UTF-8	84	VI_VN	AIX
1130	S-11	IBM-1130	84	-	主机
1164	S-11	IBM-1164	84	-	主机
1129	S-11	IBM-1129	84	-	OS/2
1258	S-11	1258	84	-	Windows

注:

1. CCSID 1392 和 5488（GB 18030）只能与 load 或 import 实用程序一起使用以将数据从 CCSID 1392 和 5488 移至 DB2 Unicode 数据库，或者从 DB2 Unicode 数据库导出至 CCSID 1392 或 5488。
2. 在 AIX 4.3 或更新版本上，代码页为 943。如果在使用 AIX 4.2 或较早版本，则代码页为 932。
3. 代码页 1394（Shift JIS X0213）仅能与 Load 或 Import 实用程序一起使用以将数据从代码页 1394 移至 DB2 Unicode 数据库，或从 DB2 Unicode 数据库导出至代码页 1394。
4. 下列各项映射为阿拉伯语国家或地区（AA）:
   - Arabic（沙特阿拉伯）
   - Arabic（伊拉克）
   - Arabic（埃及）
   - Arabic（利比亚）
   - Arabic（阿尔及利亚）
   - Arabic（摩洛哥）
   - Arabic（突尼斯）
   - Arabic（阿曼）
   - Arabic（也门）
   - Arabic（叙利亚）
   - Arabic（约旦）
   - Arabic（黎巴嫩）
   - Arabic（科威特）

- Arabic（阿拉伯联合酋长国）
- Arabic（巴林）
- Arabic（卡塔尔）

5. 下列各项映射为英语国家或地区（US）:
   - English（牙买加）
   - English（加勒比海）
6. 下列各项映射为拉丁美洲国家或地区（Lat）:
   - Spanish（墨西哥）
   - Spanish（危地马拉）
   - Spanish（哥斯达黎加）
   - Spanish（巴拿马）
   - Spanish（多米尼加共和国）
   - Spanish（委内瑞拉）
   - Spanish（哥伦比亚）
   - Spanish（秘鲁）
   - Spanish（阿根廷）
   - Spanish（厄瓜多尔）
   - Spanish（智利）
   - Spanish（乌拉圭）
   - Spanish（巴拉圭）
   - Spanish（玻利维亚）
7. 下列 Indic 脚本在 Unicode 中是受支持的: Hindi、Gujarati、Kannada、Konkani、Marathi、Punjabi、Sanskrit、Tamil 和 Telugu。
8. 代码页 950 也称为 Big5。Microsoft 代码页 950 与 IBM 代码页 950 在以下方面有所不同:

范围	描述	IBM	Microsoft	区别
X’8140’ 至 X’8DFE’	用户定义的字符	用户定义的区域	用户定义的区域	相同
X’8E40’ 至 X’A0FE’	用户定义的字符	用户定义的区域	用户定义的区域	相同
X’A140’ 至 X’A3BF’	特殊符号	系统字符	系统字符	相同
X’A3C0’ 至 X’A3E0’	控制符号	系统字符	空白	不同
X’A3E1’ 至 X’A3FE’	保留范围	空白	空白	相同
X’A440’ 至 X’C67E’	主要使用字符	系统字符	系统字符	相同
X’C6A1’ 至 X’C878’	Eten 添加的符号	系统字符	用户定义的区域	不同
X’C879’ 至 X’C8CC’	Eten 添加的符号	空白	用户定义的区域	不同
X’C8CD’ 至 X’C8D3’	Eten 添加的符号	系统字符	用户定义的区域	不同
X’C8D4’ 至 X’C8FD’	保留范围	系统字符	用户定义的区域	不同
X’C8FE’	无效的／未定义的字符	系统字符	用户定义的区域	不同
X’C940’ 至 X’F9D5’	辅助使用字符	系统字符	系统字符	相同
X’F9D6’ 至 X’F9FE’	Big-5 的 Eten 扩展	用户定义的区域	系统字符	不同

范围	描述	IBM	Microsoft	区别
X’FA40’ 至 X’FEFE’	用户定义的字符	用户定义的区域	用户定义的区域	相同
X’8181’ 至 X’8C82’	用户定义的字符	用户定义的区域	空白	不同
X’F286’ 至 X’F9A0’	IBM 选择字符	系统字符	空白	不同
总字符数		14060	13502	
用户定义的字符总数		6204	6217	
已定义的代码点总数		20264	19719	

相关任务:

- 第 340 页的『安装先前的表以便在代码页 1394 与 Unicode 之间进行转换』

亚洲字体的可用性（Linux）

IBM 提供用于 Linux 的附加字体包，包含附加的对亚洲字符双字节字符集（DBCS）的支持。对于仅安装显示特定于国家或地区的字符所需字体的某些 Linux 版本，这些字体包是必需的。

如果您在使用 DB2 安装向导或 DB2 GUI 工具（安装后）时发现缺少字符，则安装随 DB2 产品提供的必需字体，然后重新运行 **db2setup** 命令或重新启动您使用的 DB2 GUI 工具。对于 Linux 操作系统，可以在本地语言包 CD-ROM（NLPACK CD）上的 `java_fonts` 目录中找到亚洲字体。

在此目录中，有两种字型可用：Times New Roman WorldType 和 Monotype Sans Duospace WorldType。对于每种字型，都有一种特定于国家或地区的字体。下表列示了 `java_fonts` 目录中以压缩格式提供的八种字体。

字体字型	字体文件名	国家或地区
Times New Roman WT J	tnrwt_j.zip	日本及其他国家或地区
Times New Roman WT K	tnrwt_k.zip	韩国
Times New Roman WT SC	tnrwt_s.zip	中国（简体中文）
Times New Roman WT TC	tnrwt_t.zip	台湾（繁体中文）
Monotype Sans Duospace WT J	mtsansdj.zip	日本及其他国家或地区
Monotype Sans Duospace WT K	mtsansdk.zip	韩国
Monotype Sans Duospace WT SC	mtsansds.zip	中国（简体中文）
Monotype Sans Duospace WT TC	mtsansdt.zip	台湾（繁体中文）

注: 这些字体不会替换系统字体。这些字体将与 DB2 配合使用。您不能将这些字体用于一般的或无限制的销售或分发。

要安装字体:

1. 解压缩字体包。
2. 将字体包复制至 `/opt/jre/lib/fonts` 目录。如果该目录尚未存在，则需要创建它。
3. 输入以下命令：**export JAVA_FONTS=/opt/jre/lib/fonts**

注: 您可以选择将亚洲字体包复制到 BD2 安装路径的 `java` 目录中。例如: `<DB2 installation path>/java/jdk32/jre/lib/fonts` 或 `<DB2 installation path>/java/jdk64/jre/lib/fonts`。

至少需要对您所在国家或地区的每种字型安装一种字体。如果您在中国、韩国或台湾地区，则使用特定于国家或地区的版本；否则，使用字体的日语版本。如果您的系统上有空间，建议您将八种字体全部安装。

简体中文语言环境代码集

IBM AIX 以及 Linux 的某些分发已将与简体中文语言环境绑定的代码集由 GBK（代码页 1386）更改为 GB18030（代码页 5488 或 1392）。例如，在下列 AIX 版本之后，AIX 上的 Zh_CN 语言环境现在与 GB18030 代码集绑定:

- AIX V5.1.0000.0011
- 带有维护级别 2 的 AIX V5.1.0

DB2 数据库管理器本身支持 GBK 代码集，并且仅通过 Unicode 支持 GB18030 代码集。DB2 数据库管理器将把语言环境的代码集缺省为 ISO 8859-1（代码页 819），在某些操作中，还将语言环境的地域缺省为美国（US）。要克服此局限性，有两个选项:

1. 可以将语言环境的代码集从 GB18030 更改为 GBK；将地域由美国更改为中国（其地域标识是 CN，地域代码是 86）。
2. 可以使用另一个简体中文语言环境。

如果选择使用第一个选项，请发出下列命令:

```
   db2set DB2CODEPAGE=1386
db2set DB2TERRITORY=86
db2 terminate
db2stop
db2start
```

在 AIX上，如果选择使用第二个选项，请发出下列其中一个命令:

```
   export LANG=zh_CN
export LANG=ZH_CN
```

与 zh_CN 相关联的代码集是 eucCN（代码页 1383），与 ZH_CN 相关联的代码集是 UTF-8（代码页 1208）。

在 Linux 上，如果选择使用第二个选项，请发出下列其中一个命令:

```
   export LANG=zh_CN.gbk
export LANG=zh_CN
export LANG=zh_CN.utf8
```

与 zh_CN 相关联的代码集是 eucCN（代码页 1383），与 zh_CN.utf8 相关联的代码集是 UTF-8（代码页 1208）。

在 DB2 GUI 工具中显示印度字符

如果在 Linux 或 UNIX 操作系统上使用 DB2 GUI 工具时显示印度字符时有问题，则您可能没有在您的系统上安装必需的字体。

DB2 已打包以下 IBM TrueType 和 OpenType 对应印度语言字体供您使用。对于 Linux 和 UNIX 操作系统，可以在“本地语言包”CD-ROM（NLPACK CD）上的 `java_fonts` 目录中找到这些字体。

这些字体将与 DB2 配合使用。您不能将这些字体用于一般的或无限制的销售或分发:

表 99. 与 DB2 打包在一起的印度字体

字型	字形	字体文件名
Devanagari MT for IBM	中	devamt.ttf
Devanagari MT for IBM	粗体	devamtb.ttf
Tamil	中	TamilMT.ttf
Tamil	粗体	TamilMTB.ttf
Telugu	中	TeluguMT.ttf
Telugu	粗体	TeleguMTB.ttf

有关如何安装字体和修改 `font.properties` 文件的详细指示信息位于 Java 文档的 Internationalization 一节。

此外，一些 Microsoft 产品也带有可以与 GUI 工具配合使用的印度字体。

启用和禁用欧元符号支持

DB2 数据库 Linux 版、UNIX 版和 Windows 版提供对欧元货币符号的支持。已将欧元符号添加至许多代码页。

Microsoft ANSI 代码页已修改为在位置 X’80’ 包括欧元货币符号。代码页 850 已修改为将字符 DOTLESS I（在位置 X’D5’ 处）替换为欧元货币符号。DB2 内部代码页转换例程将这些修订过的代码页定义用作缺省定义以提供欧元符号支持。

但是，如果想要使用代码页转换表的非欧元定义，则在安装完成之后遵循下面的过程。

先决条件:

要替换现有外部代码页转换表文件，您可能想要在以非欧元版本覆盖当前文件之前备份这些文件。

文件位于目录 `sqllib/conv/` 中。在 UNIX 上，`sqllib/conv/` 链接至 DB2 数据库系统的安装路径。

过程:

要禁用欧元符号支持:

1. 停止 DB2 实例。
2. 下载相应的二进制格式的转换表文件:
   - 对于采用大尾数法的平台，从 ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/BigEndian/ 处下载。此 ftp 服

务器是匿名的，所以，如果正通过命令行连接，则以用户“anonymous”登录并使用电子邮件地址作为密码。在登录之后，切换至转换表目录：cd ps/products/db2/info/vr8/conv/BigEndian/

- 对于采用小尾数法的平台，从 ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/LittleEndian/ 处下载。此 ftp 服务器是匿名的，所以，如果正通过命令行连接，则以用户“anonymous”登录并使用电子邮件地址作为密码。在登录之后，切换至转换表目录：cd ps/products/db2/info/vr8/conv/LittleEndian

3. 将文件复制至 `sqllib/conv/` 目录。
4. 重新启动 DB2 实例。

代码页 819 和 1047:

对于代码页 819（ISO 8859-1 Latin 1 ASCII）和 1047（Latin 1 Open System EBCDIC），欧元替换代码页 923（ISO 8859-15 Latin 9 ASCII）和 924（Latin 9 Open System EBCDIC）分别包含欧元符号和几个新的字符。缺省情况下，DB2 数据库 Linux 版、UNIX 版和 Windows 版仍然使用这两个代码页和转换表的旧（非欧元）定义，即 819 和 1047。有两种方法来激活新 923/924 代码页和相关联的转换表:

- 创建使用新代码页的新数据库。例如，

```
DB2 CREATE DATABASE dbname USINGCODESET ISO8859-15 TERRITORY US
```

- 将 923 或 924 转换表文件从 `sqllib/conv/alt/` 目录复制至 `sqllib/conv/` 目录并将它们分别重命名为 819 或 1047。

相关概念:

- 『Character conversion』（*SQL Reference, Volume 1*）

相关参考:

- 第 315 页的『启用欧元的代码页的转换表文件』
- 第 320 页的『代码页 923 和 924 的转换表』

字符转换准则

当应用程序和数据库不使用相同的代码页时，可能需要数据转换来在应用程序和数据库代码页之间映射数据。因为映射和数据转换需要其他开销，所以，如果应用程序和数据库使用相同的代码页或标识整理顺序，则可提高应用程序性能。

在下列情况下发生字符转换:

- 当客户机或应用程序在与它访问的数据库的代码页不同的代码页中运行时。

  在接收数据的数据库服务器上进行转换。如果数据库服务器接收数据，则字符转换是从应用程序代码页到数据库代码页。如果应用程序机器接收数据，则字符转换是从数据库代码页到应用程序代码页。

- 当导入（或装入）一个文件的客户机或应用程序在与导入（或装入）的文件不同的代码页中运行时。

字符转换不对下列对象发生:

- 文件名。

• 将传送至或来自于指定了 FOR BIT DATA 属性的一列的数据，或者在其结果为 FOR BIT 或 BLOB 数据的 SQL 操作中使用的数据。
• DB2 产品或平台，没有安装受支持的至／从 EUC 或 UCS-2 的转换函数。在此情况下，应用程序接收到 SQLCODE -332（SQLSTATE 57017）错误。

数据库管理器在转换多字节代码页时使用的转换函数和转换表或 DBCS 转换 API 取决于操作系统环境。

注: 在多字节代码页之间的字符串转换，如 DBCS 与 EUC 之间，可能会增大或减小字符串的长度。另外，分配给 PC DBCS、EUC 和 UCS-2 代码集中不同字符的代码点，在对相同字符排序时也可能产生不同的结果。

扩展 UNIX 代码（EUC）代码页支持

在 C 或 C++ 应用程序中使用图形数据的主变量需要特殊考虑，包括特殊的预编译器、应用程序性能和应用程序设计问题。

在日语和繁体中文的 EUC 代码页中的很多字符需要特殊方法，用于管理数据库和客户机应用程序对图形数据（需要双字节字符）的支持。使用 UCS-2 代码集存储和处理来自这些 EUC 代码页的图形数据。

相关概念:

• 『分析在何处对联合查询求值的准则』（*《性能指南》*）

相关参考:

• 第 315 页的『启用欧元的代码页的转换表文件』
• 第 320 页的『代码页 923 和 924 的转换表』

启用欧元的代码页的转换表文件

下列各表列示已增强为支持欧元货币符号的转换表。如果想要禁用欧元符号支持，则下载标题为“转换表文件”的列中指示的转换表文件。

阿拉伯语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
864，17248	1046，9238	08641046.cnv，10460864.cnv，IBM00864.ucs
864，17248	1256，5352	08641256.cnv，12560864.cnv，IBM00864.ucs
864，17248	1200，1208，13488，17584	IBM00864.ucs
1046，9238	864，17248	10460864.cnv，08641046.cnv，IBM01046.ucs
1046，9238	1089	10461089.cnv，10891046.cnv，IBM01046.ucs
1046，9238	1256，5352	10461256.cnv，12561046.cnv，IBM01046.ucs
1046，9238	1200，1208，13488，17584	IBM01046.ucs

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
1089	1046，9238	10891046.cnv，10461089.cnv
1256，5352	864，17248	12560864.cnv，08641256.cnv，IBM01256.ucs
1256，5352	1046，9238	12561046.cnv，10461256.cnv，IBM01256.ucs
1256，5352	1200，1208，13488，17584	IBM01256.ucs

波罗的语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
921，901	1257	09211257.cnv，12570921.cnv，IBM00921.ucs
921，901	1200，1208，13488，17584	IBM00921.ucs
1257，5353	921，901	12570921.cnv，09211257.cnv，IBM01257.ucs
1257，5353	922，902	12570922.cnv，09221257.cnv，IBM01257.ucs
1257，5353	1200，1208，13488，17584	IBM01257.ucs

白俄罗斯语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
1131，849	1251，5347	11311251.cnv，12511131.cnv
1131，849	1283	11311283.cnv

Cyrillic:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
855，872	866，808	08550866.cnv，08660855.cnv
855，872	1251，5347	08551251.cnv，12510855.cnv
866，808	855，872	08660855.cnv，08550866.cnv
866，808	1251，5347	08661251.cnv，12510866.cnv
1251，5347	855，872	12510855.cnv，08551251.cnv，IBM01251.ucs
1251，5347	866，808	12510866.cnv，08661251.cnv，IBM01251.ucs
1251，5347	1124	12511124.cnv，11241251.cnv，IBM01251.ucs
1251，5347	1125，848	12511125.cnv，11251251.cnv，IBM01251.ucs

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
1251，5347	1131，849	12511131.cnv，11311251.cnv，IBM01251.ucs
1251，5347	1200，1208，13488，17584	IBM01251.ucs

爱沙尼亚语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
922，902	1257	09221257.cnv，12570922.cnv，IBM00922.ucs
922，902	1200，1208，13488，17584	IBM00922.ucs
1122，1157	1257，5353	11221257.cnv

希腊语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
813，4909	869，9061	08130869.cnv，08690813.cnv，IBM00813.ucs
813，4909	1253，5349	08131253.cnv，12530813.cnv，IBM00813.ucs
813，4909	1200，1208，13488，17584	IBM00813.ucs
869，9061	813，4909	08690813.cnv，08130869.cnv
869，9061	1253，5349	08691253.cnv，12530869.cnv
1253，5349	813，4909	12530813.cnv，08131253.cnv，IBM01253.ucs
1253，5349	869，9061	12530869.cnv，08691253.cnv，IBM01253.ucs
1253，5349	1200，1208，13488，17584	IBM01253.ucs

希伯莱语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
856，9048	862，867	08560862.cnv，08620856.cnv，IBM0856.ucs
856，9048	916	08560916.cnv，09160856.cnv，IBM0856.ucs
856，9048	1255，5351	08561255.cnv，12550856.cnv，IBM0856.ucs
856，9048	1200，1208，13488，17584	IBM0856.ucs
862，867	856，9048	08620856.cnv，08560862.cnv，IBM00862.ucs

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
862，867	916	08620916.cnv，09160862.cnv，IBM00862.ucs
862，867	1255，5351	08621255.cnv，12550862.cnv，IBM00862.ucs
862，867	1200，1208，13488，17584	IBM00862.ucs
916	856，9048	09160856.cnv，08560916.cnv
916	862，867	09160862.cnv，08620916.cnv
1255，5351	856，9048	12550856.cnv，08561255.cnv，IBM01255.ucs
1255，5351	862，867	12550862.cnv，08621255.cnv，IBM01255.ucs
1255，5351	1200，1208，13488，17584	IBM01255.ucs

拉丁语 － 1:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
437	850，858	04370850.cnv，08500437.cnv
500，1148	437	05000437.cnv，IBM00500.ucs
850，858	437	08500437.cnv，04370850.cnv
850，858	860	08500860.cnv，08600850.cnv
850，858	1114，5210	08501114.cnv，11140850.cnv
850，858	1275	08501275.cnv，12750850.cnv
860	850，858	08600850.cnv，08500860.cnv
1275	850，858	12750850.cnv，08501275.cnv

拉丁语 － 2:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
852，9044	1250，5346	08521250.cnv，12500852.cnv
1250，5346	852，9044	12500852.cnv，08521250.cnv，IBM01250.ucs
1250，5346	1200，1208，13488，17584	IBM01250.ucs

简体中文:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
837，935，1388	1200，1208，13488，17584	1388ucs2.cnv
1386	1200，1208，13488，17584	1386ucs2.cnv，ucs21386.cnv

繁体中文:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
937，835，1371	950，1370	09370950.cnv，0937ucs2.cnv
937，835，1371	1200，1208，13488，17584	0937ucs2.cnv
1114，5210	850，858	11140850.cnv，08501114.cnv

泰国语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
874，1161	1200，1208，13488，17584	IBM00874.ucs

土耳其语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
857，9049	1254，5350	08571254.cnv，12540857.cnv
1254，5350	857，9049	12540857.cnv，08571254.cnv，IBM01254.ucs
1254，5350	1200，1208，13488，17584	IBM01254.ucs

乌克兰语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
1124	1251，5347	11241251.cnv，12511124.cnv
1125，848	1251，5347	11251251.cnv，12511125.cnv

Unicode:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
1200，1208，13488，17584	813，4909	IBM00813.ucs
1200，1208，13488，17584	862，867	IBM00862.ucs
1200，1208，13488，17584	864，17248	IBM00864.ucs
1200，1208，13488，17584	874，1161	IBM00874.ucs
1200，1208，13488，17584	921，901	IBM00921.ucs
1200，1208，13488，17584	922，902	IBM00922.ucs
1200，1208，13488，17584	1046，9238	IBM01046.ucs
1200，1208，13488，17584	1250，5346	IBM01250.ucs
1200，1208，13488，17584	1251，5347	IBM01251.ucs
1200，1208，13488，17584	1253，5349	IBM01253.ucs
1200，1208，13488，17584	1254，5350	IBM01254.ucs

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
1200，1208，13488，17584	1255，5351	IBM01255.ucs
1200，1208，13488，17584	1256，5352	IBM01256.ucs
1200，1208，13488，17584	1386	ucs21386.cnv，1386ucs2.cnv

越南语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
1258，5354	1129，1163	12581129.cnv

相关概念:

- 『Character conversion』（*SQL Reference, Volume 1*）

相关任务:

- 第 313 页的『启用和禁用欧元符号支持』

代码页 923 和 924 的转换表

以下是与代码页 923 和 924 相关联的所有代码页转换表文件的列表。每个文件的格式都是 XXXXYYYY.cnv 或 ibmZZZZZ.ucs，其中 XXXXX 是源代码页号，而 YYYY 是目标代码页号。文件 ibmZZZZZ.ucs 支持代码页 ZZZZZ 和 Unicode 之间的转换。

要激活特定代码页转换表，将该转换表文件从 `sqllib/conv/alt/` 目录复制至 `sqllib/conv/` 目录，并重命名该转换表文件（如第二列中所示）。

例如，要在将 8859-1/15（Latin 1/9）客户机连接至 Windows 1252 数据库时支持欧元符号，需要复制并重命名下列代码页转换表文件:

- sqllib/conv/alt/09231252.cnv to sqllib/conv/08191252.cnv
- sqllib/conv/alt/12520923.cnv to sqllib/conv/12520819.cnv
- sqllib/conv/alt/ibm00923.ucs to sqllib/conv/ibm00819.ucs

sqlllib/conv/alt/ 目录中的 923 和 924 转换表文件	sqllib/conv/ 目录中的新名称
04370923.cnv	04370819.cnv
08500923.cnv	08500819.cnv
08600923.cnv	08600819.cnv
08630923.cnv	08630819.cnv
09230437.cnv	08190437.cnv
09230850.cnv	08190850.cnv
09230860.cnv	08190860.cnv
09231043.cnv	08191043.cnv
09231051.cnv	08191051.cnv
09231114.cnv	08191114.cnv

sqllib/conv/alt/ 目录中的 923 和 924 转换表文件	sqllib/conv/ 目录中的新名称
09231252.cnv	08191252.cnv
09231275.cnv	08191275.cnv
09241252.cnv	10471252.cnv
10430923.cnv	10430819.cnv
10510923.cnv	10510819.cnv
11140923.cnv	11140819.cnv
12520923.cnv	12520819.cnv
12750923.cnv	12750819.cnv
ibm00923.ucs	ibm00819.ucs

相关概念:

- 『Character conversion』(*SQL Reference, Volume 1*)

相关任务:

- 第 313 页的『启用和禁用欧元符号支持』

为数据库选择语言

创建数据库时，必须决定数据将以什么语言存储。创建数据库时，可指定地域和代码集。地域和代码集可以与当前操作系统设置不同。如果未在数据库创建时显式选择地域和代码集，则将使用当前语言环境创建数据库。如果正在选择代码集，则确保它可以对您将使用的语言中的所有字符进行编码。

另一选项是将数据存储在 Unicode 数据库中，这意味这不必选择特定的语言；Unicode 编码包括来自几乎世界上所有现有语言的字符。

DB2 管理服务器的语言环境设置

确保“DB2 管理服务器”实例的语言环境与 DB2 实例的语言环境兼容。否则，DB2 实例不能与“DB2 管理服务器”通信。

如果在“DB2 管理服务器”的用户概要文件中未设置 LANG 环境变量，则将使用缺省系统语言环境来启动“DB2 管理服务器”。如果未定义缺省系统语言环境，则将使用代码页 819 来启动“DB2 管理服务器”。如果 DB2 实例使用其中一种 DBCS 语言环境，而使用代码页 819 启动“DB2 管理服务器”，则该实例将无法与“DB2 管理服务器”通信。“DB2 管理服务器”的语言环境与 DB2 实例的语言环境必须兼容。

例如，在简体中文 Linux 系统中，应该在“DB2 管理服务器”的用户概要文件中设置 `LANG=zh_CN`。

相关任务:

- 『更改 DB2 界面语言（Linux 和 UNIX）』（*《DB2 服务器快速入门》*）
- 『更改 DB2 界面语言（Windows）』（*《DB2 服务器快速入门》*）

启用双向支持

使用新的“编码字符集标识”（CCSID）定义在 DB2 数据库 Linux 版、UNIX 版和 Windows 版中实现了双向格式变换。对于新的特定于双向的 CCSID，执行格式变换来代替代码页转换或在代码页转换之外执行它。要使用此支持，必须将 DB2BIDI 注册表变量设置为 YES。缺省情况下，不设置此变量。服务器对所有转换设置此变量，且只有在服务器启动后才能设置。因为存在其他检查和格式变换，将 DB2BIDI 设置为 YES 可能会影响性能。

限制:

下列限制适用:

- 若您选择对于您的客户机平台的代码页或字符串类型不适当的 CCSID，将得到意外的结果。若选择了不兼容的 CCSID（例如，对于与阿拉伯数据库的连接，选择了 Latin-1 CCSID），或者若尚未为该服务器设置 DB2BIDI，当您尝试连接时会接收到错误消息。
- Windows 操作系统上的 DB2 命令行处理器不具有双向支持。
- 对于 HOST EBCDIC 平台是客户机，而 DB2 数据库是服务器的情况，不支持 CCSID 覆盖。

当从一个阿拉伯语 CCSID 转换为另一个阿拉伯语 CCSID 时，DB2 使用以下逻辑来对 lam-alef 连字取消塑形（或展开）。当源阿拉伯语 CCSID 的“文本塑形”属性是已塑形，而目标阿拉伯语 CCSID 的“文本塑形”属性是未塑形时，就将发生取消塑形。

对 lam-alef 连字进行取消塑形的逻辑是:

1. 如果数据流的*最后*一个字符为空白字符，则将把 lam-alef 连字后面的每个字符都移至该数据流的末尾，从而为要取消塑形（展开）为它的两个构成字符：lam 和 alef 的当前 lam-alef 连字提供一个空位置。
2. 否则，如果数据流的*第一个*字符为空白字符，则将把 lam-alef 连字前面的每个字符都移至该数据流的开头，从而为要取消塑形（展开）为它的两个构成字符：lam 和 alef 的当前 lam-alef 连字提供一个空位置。
3. 否则，在数据流的开头和末尾将都没有空白字符，于是不能对 lam-alef 连字取消塑形。如果目标 CCSID 有 lam-alef 连字，则 lam-alef 连字将保持原样；否则，将使用目标 CCSID 的替换字符来替换 lam-alef 连字。

相反，当从其“文本塑形”属性设置为未塑形的阿拉伯语 CCSID 转换为其“文本塑形”属性被设置为塑形的阿拉伯语 CCSID 时，将把源 lam 和 alef 字符缩写为一个连字字符，并且在目标区域数据流的末尾插入一个空白字符。

过程:

要在非 DRDA 环境中指定特定的双向 CCSID:

- 确保 DB2BIDI 注册表变量设置为 YES。
- 选择与客户机的特征相匹配的 CCSID，并将 DB2CODEPAGE 设置为该值。
- 若您已有与该数据库的一个连接，您必须发出 TERMINATE 命令，并再次连接以使 DB2CODEPAGE 的新设置生效。

对于 DRDA 环境，若 HOST EBCDIC 平台也支持这些双向的 CCSID，您只须设置 DB2CODEPAGE 值。注意，不能在服务器数据库的 DCS 数据库目录条目的 PARMS 字段中的 BIDI 参数进一步指定同一 CCSID，否则，将发生额外的 bidi 布局转换，并且任何阿拉伯数据都将表现为进行了不正确地反向。然而，若 HOST 平台不支持这些 CCSID，则您还必须为将要连接的 HOST 数据库服务器指定一个 CCSID 覆盖。可在服务器数据库的 DCS 数据库目录条目的 PARMS 字段中使用 BIDI 参数来实现这一目的。该覆盖是必需的，因为在 DRDA 环境中，代码页转换和格式变换是由数据接收器执行的。然而，若 HOST 服务器不支持这些双向 CCSID，它将不对从 DB2 接收的数据执行格式变换。若您使用 CCSID 覆盖，DB2 客户机一样对出站数据执行格式变换。

相关概念:

- 第 326 页的『DB2 Connect 的双向支持』
- 『处理 BiDi 数据』（《*DB2 Connect 用户指南*》）

相关参考:

- 第 323 页的『特定于双向的 CCSID』
- 『一般注册表变量』（《*管理指南：实施*》）

特定于双向的 CCSID

下列双向属性是对不同平台上的双向数据作更正处理所需的:

- 文本类型
- 数字塑形
- 定向
- 文本塑形
- 对称交换

因为不同平台上的缺省值不一样，因此在将 DB2 数据从一个平台移到另一个平台时会出现问题。例如，Windows 操作系统使用 LOGICAL UNSHAPED 数据，而 z/OS 和 OS/390 通常使用 SHAPED VISUAL 数据。因而，若对双向属性没有任何支持，则从 DB2 通用数据库 z/OS 和 OS/390 版发送至 Windows 32 位操作系统工作站上的 DB2 的数据将不能正确显示。

DB2 数据库 Linux 版、UNIX 版和 Windows 版通过特殊的双向“编码字符集标识”（CCSID）来支持双向数据属性。已定义下列双向 CCSID，且对 DB2 实施了它们，如表 100 中所示。已定义 CDRA 字符串类型，如第 325 页的表 101 中所示。

表 100. 双向 CCSID

CCSID	代码页	字符串类型
420	420	4
424	424	4
856	856	5
862	862	4
864	864	5
867	862	4
916	916	5

表 100. 双向 CCSID (续)

CCSID	代码页	字符串类型
1046	1046	5
1089	1089	5
1200	1200	10
1208	1208	10
1255	1255	5
1256	1256	5
5351	1255	5
5352	1256	5
8612	420	5
8616	424	10
9048	856	5
9238	1046	5
12712	424	4
13488	13488	10
16804	420	4
17248	864	5
62208	856	4
62209	862	10
62210	916	4
62211	424	5
62213	862	5
62215	1255	4
62218	864	4
62220	856	6
62221	862	6
62222	916	6
62223	1255	6
62224	420	6
62225	864	6
62226	1046	6
62227	1089	6
62228	1256	6
62229	424	8
62230	856	8
62231	862	8
62232	916	8
62233	420	8
62234	420	9
62235	424	6
62236	856	10

表 100. 双向 CCSID (续)

CCSID	代码页	字符串类型
62237	1255	8
62238	916	10
62239	1255	10
62240	424	11
62241	856	11
62242	862	11
62243	916	11
62244	1255	11
62245	424	10
62246	1046	8
62247	1046	9
62248	1046	4
62249	1046	12
62250	420	12

表 101. CDRA 字符串类型

字符串类型	文本类型	数字塑形	定向	文本塑形	对称交换
4	可视	传递	LTR	已塑形	关闭
5	隐式	阿拉伯数字	LTR	未塑形	打开
6	隐式	阿拉伯数字	RTL	未塑形	打开
7*	可视	传递	Contextual*	未塑形连字	关闭
8	可视	传递	RTL	已塑形	关闭
9	可视	传递	RTL	已塑形	打开
10	隐式	阿拉伯数字	Contextual LTR	未塑形	打开
11	隐式	阿拉伯数字	Contextual RTL	未塑形	打开
12	隐式	阿拉伯数字	RTL	已塑形	关闭

注: *当第一个字母字符为拉丁字符时，字符串的方向为从左到右（LTR）；而当第一个字母字符为阿拉伯或希伯莱语字符时，字符串的方向为从右到左（RTL)。字符未塑形，但是保留了 LamAlef 连字，并且没有将它分开。

相关概念:

- 第 326 页的『DB2 Connect 的双向支持』

相关任务:

- 第 322 页的『启用双向支持』

DB2 Connect 的双向支持

当在 DB2 Connect 与服务器上的数据库之间交换数据时，通常是接收方对入局数据执行转换。除通常的代码页转换外，同样的约定通常也适用于双向格式变换。DB2 Connect 能够对它准备发送至服务器数据库的数据和从服务器数据库接收到的数据执行双向格式变换。

为了 DB2 Connect 能够对服务器数据库的出局数据执行双向格式变换，必须替换服务器数据库的双向 CCSID。可在服务器数据库的 DCS 数据库目录条目的 PARMS 字段中使用 BIDI 参数来实现这一目的。

注: 如果希望 DB2 Connect 对准备发送至 DB2 主机或 iSeries 数据库的数据执行格式变换，即使不必覆盖其 CCSID，也必须将 BIDI 参数添加至 DCS 数据库目录的 PARMS 字段。在这种情况下，应提供的 CCSID 是缺省 DB2 主机或 iSeries 数据库 CCSID。

将 BIDI 参数指定为 PARMS 字段中的第九个参数，同时指定双向 CCSID，希望替换缺省的服务器数据库双向 CCSID:

```
",,,,,,,,BIDI=xyz"
```

其中 *xyz* 是 CCSID 覆盖

注: 要使 BIDI 参数生效，必须将注册表变量 DB2BIDI 设置为 YES。

为了最好地描述如何使用此功能部件，举例如下。

假设您有一个运行 CCSID 62213（双向字符串类型 5）的希伯莱语版 DB2 的客户机，而您又想访问运行 CCSID 00424（双向字符串类型 4）的 DB2 主机或 iSeries 数据库。但您知道包含在 DB2 主机或 iSeries 数据库中的数据基于 CCSID 08616（双向字符串类型 6）。

这样，就会有两个问题：第一个问题是 DB2 主机或 iSeries 数据库不能区分 CCSID 00424 和 08616 的双向字符串类型。第二个问题是 DB2 主机或 iSeries 数据库不识别 DB2 客户机 CCSID（62213）。它只支持 CCSID 00862，它与 CCSID 62213 基于相同的代码页。

需要确保发送至 DB2 主机或 iSeries 数据库的数据以双向字符串类型 6 格式开头，并且要让 DB2 Connect 知道它必须对从 DB2 主机或 iSeries 数据库接收到的数据执行双向变换。需要对 DB2 主机或 iSeries 数据库使用以下编目命令:

```
db2 catalog dcs database nydb1 as telaviv parms ",,,,,,,,BIDI=08616"
```

此命令告诉 DB2 Connect 使用 CCSID 08616 覆盖 DB2 主机或 iSeries 数据库的 CCSID 00424。此覆盖包括下列处理:

1. DB2 Connect 连接至使用 CCSID 00862 的 DB2 主机或 iSeries 数据库。
2. DB2 Connect 对它准备发送至 DB2 主机或 iSeries 数据库的数据执行双向格式变换。即从 CCSID 62213（双向字符串类型 5）变换为 CCSID 62221（双向字符串类型 6）。
3. DB2 Connect 对它从 DB2 主机或 iSeries 数据库*接收*到的数据执行双向格式变换。即从 CCSID 08616（双向字符串类型 6）变换为 CCSID 62213（双向字符串类型 5）。

注: 有些情况下，使用双向 CCSID 可能造成 SQL 查询本身被修改，结果使 DB2 服务器不能识别该 SQL 查询。特别是，当可以使用另一种字符串类型时，应避免使用 IMPLICIT CONTEXTUAL CCSID 和 IMPLICIT RIGHT-TO-LEFT CCSID。若 SQL 查询包括引用字符串，则 CONTEXTUAL CCSID 可能产生不可预测的结果。避免在 SQL 语句中使用引用字符串；尽可能使用主变量。

若某个特定的双向 CCSID 导致不能按照下列建议改正的问题，则将 DB2BIDI 设置为 NO。

相关概念:

- 『处理 BiDi 数据』（*《DB2 Connect 用户指南》*）

相关参考:

- 第 323 页的『特定于双向的 CCSID』

整理顺序

数据库管理器使用*整理顺序*来比较字符数据。这是对一组字符的排序，确定某个字符的 ASCII 码是大于、小于还是等于另一个字符的 ASCII 码。

注: 用 FOR BIT DATA 属性定义的字符串数据和 BLOB 数据使用二进制排序顺序进行排序。

例如，可使用整理顺序来指示特定字符的小写与大写的 ASCII 码在排序时视为相等。

数据库管理器允许使用定制的整理顺序来创建数据库。以下各节帮助您确定和实现数据库的特定整理顺序。

数据库管理器允许使用定制的整理顺序来创建数据库。对于 Unicode 数据库，在『DB2 数据库中的 Unicode 实现』主题中描述了各种受支持的整理顺序。以下各节帮助您确定和实现数据库的特定整理顺序。

数据库中的每个单字节字符在内部表示为 0 与 255 之间（采用十六进制表示法，在 X'00' 与 X'FF' 之间）的唯一数字。此数字又称字符的*代码点*；将数字指定为字符的集合统称为*代码页*。整理顺序是代码点与排序顺序中每个字符的期望位置之间的映射。该位置的数字值在整理顺序中称为字符的*权重*。在最简单的整理顺序中，权重与代码点完全相同。这称为*标识顺序*。

例如，假定字符 B 和 b 分别具有代码点 X'42' 和 X'62'。如果（根据整理顺序表）它们都具有排序权重 X'42'（B），则它们的整理顺序相同。如果 B 的排序权重为 X'9E'，而 b 的排序权重为 X'9D'，则排序时 b 将在 B 之前。整理顺序表指定每个字符的权重。该表不同于代码页，后者指定每个字符的代码点。

考虑以下示例。ASCII 字符 A 到 Z 由 X'41' 到 X'5A' 表示。要描述用来对这些字符连续排序（没有干扰字符）的整理顺序，可以这样写：X'41'，X'42'，... X'59'，X'5A'。

多字节字符的十六进制值还用作权重。例如，假定双字节字符 A 和 B 的代码点分别为 X'8260' 和 X'8261'，则使用 X'82'、X'60' 和 X'61' 的整理权重来根据这两个字符的代码点来进行排序。

整理顺序中的权重不一定是唯一的。例如，可对同一字母的大小写指定相同权重。

如果整理顺序提供所有 256 个代码点的权重，指定整理顺序将会非常简单。每个字符的权重可使用字符的代码点来确定。

在所有情况下，DB2 数据库使用在创建数据库时指定的整理表。如果想要多字节字符以它们出现在代码点表中的顺序进行排序，在创建数据库时必须将 IDENTITY 指定为整理顺序。

注: 对于 Unicode 数据库，在『DB2 数据库中的 Unicode 实现』主题中描述了各种受支持的整理顺序。

一旦定义了整理顺序，就将用该整理顺序执行该数据库将来的所有字符比较。除了定义为 FOR BIT DATA 或 BLOB 数据的字符数据之外，将对所有 SQL 比较和 ORDER BY 子句使用整理顺序，在设置索引和统计信息时也会使用整理顺序。

在下列情况下可能会出现潜在问题:

- 应用程序将数据库中已排序的数据与应用程序的数据合并在一起，应用程序数据是使用不同的整理顺序排序的。
- 应用程序将一个数据库中已排序的数据与另一个数据库中已排序的数据合并在一起，但这两个数据库的整理顺序不同。
- 应用程序对排序的数据所作的假定不符合相关的整理顺序。例如，数字排在字母之前对特定的整理顺序可能适用也可能不适用。

最后一点要记住的是，对字符代码点进行直接比较所得到的任何排序的结果将只与使用等同的整理顺序排序的查询结果匹配。

相关概念:

- 『Character comparisons based on collating sequences』（*Developing SQL and External Routines*）
- 『Character conversion』（*SQL Reference, Volume 1*）
- 第 333 页的『DB2 数据库 Linux 版、UNIX 版和 Windows 版中的 Unicode 实施』

整理泰国语字符

泰国语包含特殊元音（“前导元音”）、音调符号和其他不按顺序排序的特殊字符。

限制:

必须要么以泰国语语言环境和代码集创建数据库，要么创建 Unicode 数据库。

过程:

当使用泰国语和相应的代码集创建数据库时，在 CREATE DATABASE 命令上使用 COLLATE USING NLSCHAR 子句。当创建 Unicode 数据库时，在 CREATE DATABASE 命令上使用 COLLATE USING UCA400_LTH 子句。

相关概念:

- 第 327 页的『整理顺序』

相关参考:

- 『CREATE DATABASE command』（*Command Reference*）

基于地域代码的日期和时间格式

用字符串表示的日期和时间格式是与应用程序的地域代码相关的日期时间值的缺省格式。当对程序进行预编译或将其绑定至数据库时，可指定 DATETIME 格式选项来覆盖此缺省格式。

以下是日期和时间的输入和输出格式的描述:

- 输入时间格式
  - 无缺省输入时间格式。
  - 对于所有地域代码，允许所有时间格式作为输入。
- 输出时间格式
  - 缺省输出时间格式等于本地时间格式。
- 输入日期格式
  - 无缺省输入日期格式。
  - 在日期的本地格式与 ISO、JIS、EUR 或 USA 日期格式冲突之处，将把本地格式视为日期输入格式。例如，查看表 102 中的 UK 条目。
- 输出日期格式
  - 在表 102 中显示了缺省输出日期格式。

  注: 表 102 还显示了各种地域代码的字符串格式的列表。

表 102. 基于地域代码的日期和时间格式

地域代码	本地日期格式	本地时间格式	缺省输出日期格式	输入日期格式
355 阿尔巴尼亚	yyyy-mm-dd	JIS	LOC	LOC，USA，EUR，ISO
785 阿拉伯	dd/mm/yyyy	JIS	LOC	LOC，EUR，ISO
001 澳大利亚（1）	mm-dd-yyyy	JIS	LOC	LOC，USA，EUR，ISO
061 澳大利亚	dd-mm-yyyy	JIS	LOC	LOC，USA，EUR，ISO
032 比利时	dd/mm/yyyy	JIS	LOC	LOC，EUR，ISO
055 巴西	dd.mm.yyyy	JIS	LOC	LOC，EUR，ISO
359 保加利亚	dd.mm.yyyy	JIS	EUR	LOC，USA，EUR，ISO
001 加拿大	mm-dd-yyyy	JIS	USA	LOC，USA，EUR，ISO
002 加拿大（法语区）	dd-mm-yyyy	ISO	ISO	LOC，USA，EUR，ISO
385 克罗地亚	yyyy-mm-dd	JIS	ISO	LOC，USA，EUR，ISO
042 捷克共和国	yyyy-mm-dd	JIS	ISO	LOC，USA，EUR，ISO
045 丹麦	dd-mm-yyyy	ISO	ISO	LOC，USA，EUR，ISO

表 102. 基于地域代码的日期和时间格式 (续)

地域代码	本地日期格式	本地时间格式	缺省输出日期格式	输入日期格式
358 芬兰	dd/mm/yyyy	ISO	EUR	LOC，EUR，ISO
389 FYR 马其顿	dd.mm.yyyy	JIS	EUR	LOC，USA，EUR，ISO
033 法国	dd/mm/yyyy	JIS	EUR	LOC，EUR，ISO
049 德国	dd/mm/yyyy	ISO	ISO	LOC，EUR，ISO
030 希腊	dd/mm/yyyy	JIS	LOC	LOC，EUR，ISO
036 匈牙利	yyyy-mm-dd	JIS	ISO	LOC，USA，EUR，ISO
354 冰岛	dd-mm-yyyy	JIS	LOC	LOC，USA，EUR，ISO
091 印度	dd/mm/yyyy	JIS	LOC	LOC，EUR，ISO
972 以色列	dd/mm/yyyy	JIS	LOC	LOC，EUR，ISO
039 意大利	dd/mm/yyyy	JIS	LOC	LOC，EUR，ISO
081 日本	mm/dd/yyyy	JIS	ISO	LOC，USA，EUR，ISO
082 韩国	mm/dd/yyyy	JIS	ISO	LOC，USA，EUR，ISO
001 拉丁美洲 (1)	mm-dd-yyyy	JIS	LOC	LOC，USA，EUR，ISO
003 拉丁美洲	dd-mm-yyyy	JIS	LOC	LOC，EUR，ISO
031 荷兰	dd-mm-yyyy	JIS	LOC	LOC，USA，EUR，ISO
047 挪威	dd/mm/yyyy	ISO	EUR	LOC，EUR，ISO
048 波兰	yyyy-mm-dd	JIS	ISO	LOC，USA，EUR，ISO
351 葡萄牙	dd/mm/yyyy	JIS	LOC	LOC，EUR，ISO
086 中国	mm/dd/yyyy	JIS	ISO	LOC，USA，EUR，ISO
040 罗马尼亚	yyyy-mm-dd	JIS	ISO	LOC，USA，EUR，ISO
007 俄罗斯	dd/mm/yyyy	ISO	LOC	LOC，EUR，ISO
381 塞尔维亚／蒙的内哥罗	yyyy-mm-dd	JIS	ISO	LOC，USA，EUR，ISO
042 斯洛伐克	yyyy-mm-dd	JIS	ISO	LOC，USA，EUR，ISO
386 斯洛文尼亚	yyyy-mm-dd	JIS	ISO	LOC，USA，EUR，ISO
034 西班牙	dd/mm/yyyy	JIS	LOC	LOC，EUR，ISO
046 瑞典	dd/mm/yyyy	ISO	ISO	LOC，EUR，ISO
041 瑞士	dd/mm/yyyy	ISO	EUR	LOC，EUR，ISO
088 台湾	mm-dd-yyyy	JIS	ISO	LOC，USA，EUR，ISO

表 102. 基于地域代码的日期和时间格式 (续)

地域代码	本地日期格式	本地时间格式	缺省输出日期格式	输入日期格式
066 泰国 (2)	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
090 土耳其	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
044 英国	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
001 美国	mm-dd-yyyy	JIS	USA	LOC, USA, EUR, ISO
084 越南	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO

注:

1. 将地域代码 001 指定给使用缺省 C 语言环境的国家或地区。
2. 佛教纪元的 yyyy 等价于罗马教的 yyyy + 543 年(限于泰国)。

相关参考:

- 『BIND command』(*Command Reference*)
- 『PRECOMPILE command』(*Command Reference*)

Unicode 字符编码

Unicode 字符编码标准是固定长度的字符编码方案，它包含了世界上几乎所有现用语言的字符。

有关 Unicode 的信息可在最新版本的 The Unicode Standard 一书中找到，并且可从“Unicode 协会”(Unicode Consortium)Web 站点(www.unicode.org)中找到。

Unicode 使用两种编码格式: 8 位和 16 位。缺省编码格式是 16 位，即每个字符是 16 位(两个字节)宽，并且通常显示为 U+hhhh，其中 hhhh 是字符的十六进制代码点。生成的 65000+ 代码元素足以用于编码世界上主要语言的大多数字符，Unicode 标准还提供了一种扩展机制，允许编码一百多万个字符。扩展机制使用一对高位和低位代用字符来对扩展字符或补充字符进行编码。第一个(或高位)代用字符具有 U+D800 和 U+DBFF 之间的代码值，而第二个(或低位)代用字符具有 U+DC00 和 U+DFFF 之间的代码值。

UCS-2

“国际标准组织”(ISO)和“国际电工委员会”(IEC)标准 10646(ISO/IEC 10646)指定了“通用多八位字节编码字符集”(UCS)，该编码字符集有一个 16 位(双字节)版本(UCS-2)和一个 32 位(四字节)版本(UCS-4)。UCS-2 相当于没有代用字符的 Unicode 16 位格式。UCS-2 可以对 Unicode 版本 3.0 指令表中定义的所有(16 位)字符进行编码。两个 UCS-2 字符 – 一个高位代用字符后面跟随一个低位代用字符 – 需要对从 Unicode 版本 3.1 开始引入的每个新补充的字符进行编码。这些补充字符在原始的 16 位“基本多语言位面”(BMP 或位面 0)外部定义。

UTF-8

对于面向字节基于 ASCII 的应用程序和文件系统，16 位 Unicode 字符是引起问题的主要因素。例如，不明白 Unicode 的应用程序可能会将大写字符“A”（U+0041）的 8 个前导零位误解为单字节 ASCII NULL 字符。

UTF-8（UCS 变换格式 8）是一种算法变换，它将定长 Unicode 字符变换为变长 ASCII 安全的字节字符串。在 UTF-8 中，ASCII 和控制字符由通常的单字节代码表示，但其他字符变为双字节或更多字节。UTF 8 可以对非补充和补充字符进行编码。

UTF 16

ISO/IEC 10646 还定义了一种扩展技术，以使用两个 UCS-2 字符来编码某些 UCS-4 字符。此扩展技术称为 UTF-16，它相当于有代用字符的 Unicode 16 位编码格式。总之，UTF-16 字符指令集由所有 UCS-2 字符和可通过代用字符访问的附加的一百万字符构成。

当将 16 位 Unicode 字符序列化为字节时，某些处理器将最重要的字节放置在初始位置（称为大尾数法排序），而其他的处理器则首先放置最不重要的字节（称为小尾数法排序）。Unicode 的缺省字节排序是大尾数法。

使用 UTF-8 格式的每个 UTF-16 字符的字节数可根据表 103 来确定。

表 103. UTF-8 的位分布

代码值 （二进制）	UTF-16 （二进制）	第一个字节 （二进制）	第二个字节 （二进制）	第三个字节 （二进制）	第四个字节 （二进制）
00000000 0xxxxxxx	00000000 0xxxxxxx	0xxxxxxx			
00000yyy yyxxxxxx	00000yyy yyxxxxxx	110yyyyy	10xxxxxx		
zzzzyyyy yyxxxxxx	zzzzyyyy yyxxxxxx	1110zzzz	10yyyyyy	10xxxxxx	
uuuuu zzzzyyyy yyxxxxxx	110110ww wwzzzzyy 110111yy yyxxxxxx	11110uuu （其中 uuuuu = wwww+1）	10uuzzzz	10yyyyyy	10xxxxxx

在以上每一项中，u、w、x、y 和 z 串都是字符的位表示法。例如，U+0080 变换为二进制中的 11000010 10000000，而代用字符对 U+D800 U+DC00 变为二进制中的 11110000 10010000 10000000 10000000。

相关概念:

- 第 336 页的『数据类型的 Unicode 处理』
- 第 333 页的『DB2 数据库 Linux 版、UNIX 版和 Windows 版中的 Unicode 实施』
- 第 339 页的『Unicode 文字』

相关任务:

- 第 337 页的『创建 Unicode 数据库』

DB2 数据库 Linux 版、UNIX 版和 Windows 版中的 Unicode 实施

DB2 数据库 Linux 版、UNIX 版和 Windows 版支持 UTF-8 和 UCS-2。

当创建 Unicode 数据库时，CHAR、VARCHAR、LONG VARCHAR 和 CLOB 数据以 UTF-8 格式存储，而 GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC 和 DBCLOB 数据以 UCS-2 大尾数法格式存储。

在版本 7.2 修订包 4 之前的 DB2 产品版本中，DB2 将代用字符对中的两个字符看作两个独立的 Unicode 字符。因此，将该代用字符对从 UTF-16/UCS-2 变换为 UTF-8 会生成两个三字节序列。从 DB2 通用数据库 版本 7.2 修订包 4 开始，DB2 在 UTF-16/UCS-2 和 UTF-8 之间变换时识别代用字符对，因此一对 UTF-16 代用字符将会变为一个 UTF-8 四字节序列。在其他用法中，DB2 继续将代用字符对看作两个独立的 UCS-2 字符。只要知道如何区分补充字符与非补充字符，就可以安全地将补充字符存储在 DB2 Unicode 数据库中。

DB2 将每个 Unicode 字符看作一个单独字符，其中包括诸如 COMBINING ACUTE ACCENT 字符（U+0301）的那些（非空格）字符。因此，DB2 不会将字符 LATIN SMALL LETTER A WITH ACUTE（U+00E1）识别为与后跟字符 COMBINING ACUTE ACCENT（U+0301）的字符 LATIN SMALL LETTER A（U+0061）严格等价。

UCS-2 Unicode 数据库的缺省整理顺序是 IDENTITY，它根据代码点对字符进行排序。因此，缺省情况下，将根据代码点来排序和比较所有 Unicode 字符。对于非补充 Unicode 字符，当使用 UTF-8 和 UCS-2 编码时，它们的二进制整理顺序是相同的。但是，如果任何补充字符需要一对代用字符才能进行编码，则在使用 UTF-8 编码时将把该字符向末尾整理，但是，使用 UCS-2 编码时将把同一字符向中间的某处整理，并且可以将它的两个代用字符分开。原因在于当以 UTF-8 编码时，扩展字符具有四字节二进制代码值 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx，它大于 UTF-8 编码 U+FFFF，即 X'EFBFBF'。但是在 UCS 中，相同的补充字符被编码为一对 UCS-2 高位和低位代用字符，并且具有二进制格式 1101 1000 xxxx xxxx 1101 1100 xxxx xxxx，它小于 UCS-2 编码 U+FFFF。

还可以使用 IDENTITY_16BIT 整理选项创建 Unicode 数据库。IDENTITY_16BIT 整理器按照 Unicode Technical Report #26 中所指定的那样实施 CESU-8 *UTF-16 的兼容性编码方案：8 位算法*，Unicode Technical Report #26 可在 Unicode Technical Consortium（Unicode 技术协会）Web 站点（www.unicode.org）上获得。除了 Unicode 补充字符之外，CESU-8 是与 UTF-8 完全相同的二进制字符，那些补充字符是在 16 位“基本多语言位面”（BMP 或位面 0）之外定义的。在 UTF-8 编码中，补充字符由一个四字节的序列表示，但 CESU-8 中的相同字符需要两个三字节的序列。使用 IDENTITY_16BIT 整理选项对字符和图形数据会生成相同的整理顺序。

对于 Unicode 数据库，DB2 UDB 版本 8.2 支持三个新的整理顺序关键字：UCA400_NO、UCA400_LSK 和 UCA400_LTH。UCA400_NO 整理器根据将规范化隐式设置为打开的“Unicode 标准”版本 4.00 来实现 UCA（Unicode 整理算法）。UCA400_LSK 和 UCA400_LTH 整理器还实现了 UCA V4.00。UCA400_LSK 按适当的顺序对所有斯洛伐克语字符进行排序，而 UCA400_LTH 按照皇家泰国语字典顺序对

所有泰国语字符进行排序。有关 UCA 的详细信息可以在 Unicode Consortium Web 站点（www.unicode.org）上的 Unicode Technical Standard #10 中找到。

所有与文化相关的参数，如日期或时间格式、十进制分隔符及其他参数等等，都是以客户机的当前地域为基础。

Unicode 数据库允许与 DB2 支持的所有代码页连接。数据库管理器自动地在客户机的代码页与 Unicode 之间对字符和图形字符串执行代码页转换。

任何客户机均受字符编码系统、输入方法和它的环境所支持的字体限制，但是 UCS-2 数据库本身接受并存储所有 UCS-2 字符。因此，每个客户机通常使用 UCS-2 字符的一个子集，但数据库管理器允许使用 UCS-2 字符的整个编码系统。

当字符由本地代码页转换为 Unicode 时，字节数可能增多。在版本 8 之前，根据 SQL 语句的语义，已经将字符数据标记为在客户机的代码页中编码，并且数据库服务器可能在客户机的代码页中处理了整个语句。这种处理可能导致了数据的潜在扩展。从版本 8 开始，一旦将 SQL 语句输入到数据库服务器中，它仅对数据库服务器的代码页进行操作。在此情况下，没有大小更改。然而，为某些字符串函数指定字符串单元可能会导致内部代码页转换。如果发生内部代码页转换，则数据字符串的大小可能更改。

AIX、UNIX 和 Linux 分发和代码页

较新版本的 AIX、某些 UNIX 平台和许多 Linux 分发产品都使用 Unicode（UTF-8）作为缺省代码页，而不是使用传统的非 Unicode 代码页。如果在系统上升级了操作系统，并且升级操作更改了缺省代码页，则:

- 由于已修改缺省的活动代码页，因此，以前正常运行的应用程序可能会失败。
- 除非在创建新数据库时显式地指定代码页，否则操作系统升级后创建的任何新数据库都将使用 UTF-8 Unicode 代码页来创建。所有现有数据库都将保持它们的原始代码页设置；即，在数据库创建期间建立的设置。

要确定 Linux 上运行的系统的活动代码页，请运行:

```
locale
```

运行此命令时，并非显示的所有信息都是重要的或者有意义的，但是，DB2 数据库管理器按列示顺序使用下列各项来确定活动代码页:

- LC_ALL
- LC_CTYPE
- LANG

要确定数据库所使用的代码页，请运行:

```
db2 get db cfg for <database name>
```

然后，检查“Database code page”参数的值。

代码页／CCSID 号码

在 IBM 内部，已将 UCS-2 代码页注册为代码页 1200，它的字符集在不断增加；即，当将新字符添加至代码页时，代码页号码不变。代码页 1200 始终引用 Unicode 的当前版本。

UCS 标准的特定版本由 Unicode 2.0 和 ISO/IEC 10646-1 定义，在 IBM 内部也已将它注册为 CCSID 13488。此 CCSID 已在 DB2 内部使用，用于存储 IBM eucJP（日本）和 IBM eucTW（中国台湾）数据库中的图形字符串数据。CCSID 13488 和代码页 1200 都引用 UCS-2，除了它们的"双字节"(DBCS) 空格的值不同外，其处理方式是相同的:

CP/CCSID	单字节（SBCS）空间	双字节（DBCS）空间
1200	不适用	U+0020
13488	不适用	U+3000

注: 在 UCS-2 数据库中，U+3000 没有任何特殊意义。

关于转换表，由于代码页 1200 是 CCSID 13488 的超集，所以对它们使用了相同的（超集）表。

在 IBM 内部，已将 UTF-8 注册为具有增长的字符集的 CCSID 1208（有时也称为代码页 1208）。当向标准集中添加新字符时，此号码（1208）不会改变。

MBCS 代码页号码为 1208，它是数据库代码页号码以及该数据库中字符串的代码页。UCS-2 的双字节代码页号码为 1200，它是数据库中的图形字符串数据的代码页。

泰国语和 Unicode 整理算法的区别

在具有 NLSCHAR 整理选项的"泰国语业界标准"(TIS）TIS620-1（代码页 874）泰国语数据库中使用的整理算法与具有 UCA400_LTH 整理选项的 Unicode 数据库中使用的整理算法类似，但是不完全相同。它们之间的区别如下:

- 当对 TIS620-1 数据进行排序时，每个字符只有一种字形，并且在整理期间该字形用来与另一个字符的字形进行比较。当对 Unicode 数据进行排序时，每个字符都有几种字形，并且在整理期间可以使用该字符的所有字形。
- 当对 TIS620-1 数据进行排序时，空格字符 X'20'、连字符 X'2D' 和句点字符 X'2E' 的字形小于所有泰国语字符的字形。但是，当对 Unicode 数据进行排序时，这三个字符被认为是标点标志；并且仅当进行比较的两个字符串中的所有其他字符都相同时才用于比较。
- 当 TIS620-1 数据库中的 Paiyannoi 字符 X'CF' 和 Maiyamok 字符 X'E6' 跟在其他泰国语字符后面时，就将它们认为是标点符号，当它们出现在字符串开头时，就认为它们是正常字符，并且具有它们自己的字形。Unicode 数据库中的这两个字符（分别为 U+0E2F 和 U+0E46）始终被视为标点符号，并且在进行比较的两个字符串中的所有其他字符都相同时才用于比较。

有关泰国语字符的更多信息，可以在 Thai of the Unicode Standard 一书（版本 4.0）（ISBN 0-321-18578-1）的第 10.1 章中找到。

相关概念:

- 第 331 页的『Unicode 字符编码』
- 第 336 页的『数据类型的 Unicode 处理』
- 第 339 页的『Unicode 文字』

相关任务:

- 第 337 页的『创建 Unicode 数据库』

相关参考:

- 『Character strings』（*SQL Reference, Volume 1*）

数据类型的 Unicode 处理

DB2 数据库 Linux 版、UNIX 版和 Windows 版支持的所有数据类型在 UCS-2 数据库中也受支持。特别是，UCS-2 数据库支持图形字符串数据，并以 UCS-2／Unicode 存储。每个客户机（包括 SBCS 客户机）与 UCS-2 数据库连接时，可使用 UCS-2／Unicode 格式的图形字符串数据类型。

UCS-2 数据库同任何 MBCS 数据库一样，其字符串数据以字节数计。当使用 UTF-8 格式的字符串数据时，不应当认为每个字符都是单字节。在多字节 UTF-8 编码中，每个 ASCII 字符都是单字节，但每个非 ASCII 字符为两个至四个字节。当定义 CHAR 字段时应考虑这点。取决于 ASCII 与非 ASCII 字符的比率，一个大小为 *n* 字节的 CHAR 字段可包含 *n*/4 到 *n* 之间的任意多个字符。

对图形字符串 UCS-2 数据类型使用字符串 UTF-8 编码也对总存储器需求有影响。对于大多数字符是 ASCII 但其间插入一些非 ASCII 字符的情况，存储 UTF-8 数据可能是比较好的替代方法，因为存储器需求接近每个字符一个字节。另一方面，在大多数字符都是扩充为三字节或四字节 UTF-8 序列的非 ASCII 字符（例如，表意字符）的情况下，UCS-2 图形字符串格式可能是更好的备用格式，这是因为每个三字节 UTF-8 序列都成为 16 位的 UCS-2 字符，而每个四字节 UTF-8 序列都成为两个 16 位的 UCS-2 字符。

在 MBCS 环境中处理字符串的 SQL 函数，如 SUBSTR、POSSTR、MAX 和 MIN 等，是对“字节”数而不是“字符”数进行运算的。该行为在 UCS-2 数据库中也一样，但当为 UCS-2 数据库指定偏移量和长度时应格外小心，因为始终是在数据库代码页的上下文中定义这些值。也就是说，对于 UCS-2 数据库，应在 UTF-8 中定义这些值。因为一些单字节字符在 UTF-8 格式下需要多个字节，因此对单字节数据库有效的 SUBSTR 索引可能对 UCS-2 数据库无效。若指定了不正确的索引，则将返回 SQLCODE -191（SQLSTATE 22504）。

注: 并非所有对字符串进行运算的 SQL 函数都被限制为处理“字节数”。CHARACTER_LENGTH、LENGTH、LOCATE、POSITION 和 SUBSTRING 函数包括一个参数，它允许您指定一组预定义的字符串单元。这表示这些函数可以使用指定的单位而不是字节数或双字节数来处理字符串。

在用户程序中，（C 语言中的）`char` 数据类型支持 SQL CHAR 数据类型。在用户程序中，`sqldbchar` 支持 SQL GRAPHIC 数据类型。注意，对于 UCS-2 数据库，`sqldbchar` 数据始终使用大尾数法（高字节在前）格式。在应用程序连接至 UCS-2 数据库时，DB2 将在应用程序代码页与 UTF-8 之间转换字符字符串数据，并在应用程序图形代码页与 UCS-2 之间转换图形字符串数据。

当将数据从 Unicode 数据库检索至不使用 SBCS、EUC 或 Unicode 代码页的应用程序时，将对每个填充到图形列中的空白返回已定义的替换字符。DB2 用 ASCII 空白（U+0200）填充定长 Unicode 图形列，这是一个在纯 DBCS 代码页中没有等价字符的字符。因此，进行检索时，将把填充图形列时使用的每个 ASCII 空白都转换为替换字

符。同样，在 DATE、TIME 或 TIMESTAMP 字符串中，当从 Unicode 数据库检索到不使用 SBCS、EUC 或 Unicode 代码页的应用程序时，还将把任何不具有纯 DBCS 等价字符的 SBCS 字符转换为替换字符。

注: 在版本 8 之前，始终假定图形字符串数据使用 UCS-2。为了向依赖于 DB2 的先前行为的应用程序提供向后兼容性，已引入注册表变量 DB2GRAPHICUNICODESERVER。其缺省值是 OFF。将此变量的值更改为 ON 将导致 DB2 使用其先前行为并假定图形字符串数据始终使用 UCS-2。另外，DB2 服务器将检查客户机上运行的 DB2 的版本，如果客户机正在运行 DB2 UDB 版本 7，则服务器将模拟 DB2 通用数据库 版本 7 的行为。

相关概念:

- 第 331 页的『Unicode 字符编码』
- 第 333 页的『DB2 数据库 Linux 版、UNIX 版和 Windows 版中的 Unicode 实施』

创建 Unicode 数据库

缺省情况下，用创建数据库的应用程序的代码页创建数据库。因此，如果从 Unicode（UTF-8）客户机创建数据库，数据库将创建为 Unicode 数据库。或者，可显式地将 “UTF-8” 指定为 CODESET 名称，并使用 DB2 数据库 Linux 版、UNIX 版和 Windows 版支持的任何有效 TERRITORY 代码。

在 BD2 数据库管理器的将来发行版中，无论应用程序代码页为多少，在创建数据库时都会将缺省代码集更改为 UTF-8。

过程:

要使用美国地域代码来创建 Unicode 数据库:

```
DB2 CREATE DATABASE dbname USING CODESET UTF-8 TERRITORY US
```

要使用 **sqlecrea** API 创建 Unicode 数据库，应在 *sqledbterritoryinfo* 中相应地设置值。例如，将 SQLDBCODESET 设置为 `UTF-8`，将 SQLDBLOCALE 设置为任何有效的地域代码（例如 `US`）。

相关概念:

- 第 333 页的『DB2 数据库 Linux 版、UNIX 版和 Windows 版中的 Unicode 实施』

相关任务:

- 第 338 页的『将非 Unicode 数据库转换为 Unicode 数据库』

相关参考:

- 『sqlecrea API - Create database』（*Administrative API Reference*）
- 『CREATE DATABASE command』（*Command Reference*）
- 第 291 页的『受支持的地域代码和代码页』

将非 Unicode 数据库转换为 Unicode 数据库

在某些情况下，可能需要将现有非 Unicode 数据库转换为 Unicode 数据库。例如，因为 XML 列仅在 Unicode 数据库中受支持，如果想要将 XML 列添加至现有非 Unicode 数据库，则在添加 XML 列之前，需要将该数据库转换为 Unicode 数据库。

先决条件:

必须具有足够的可用磁盘空间以从非 Unicode 数据库中导出数据。此外，如果未在复用现有表空间，则还需要足够的可用磁盘空间来为数据创建新的表空间。

限制:

XML 数据只能存储在使用 UTF-8 代码集定义的单一分区数据库中。

过程:

下列步骤演示了如何将现有非 Unicode 数据库转换为 Unicode 数据库:

1. 使用 **db2move** 命令导出数据:

   ```
   cd <export-dir>
   db2move sample export
   ```

   其中 <export-dir> 是要将数据导出至的目录，而 SAMPLE 是现有数据库名称。
2. 使用 **db2look** 命令为现有数据库生成 DDL 脚本:

   ```
   db2look -d sample -e -o unidb.ddl -l -x -f
   ```

   其中 SAMPLE 是现有数据库名称，而 unidb.ddl 是生成的 DDL 脚本的文件名。-l 选项为用户定义的表空间、数据库分区组和缓冲池生成 DDL，-x 选项生成授权 DDL，而 -f 选项为数据库配置参数生成更新命令。
3. 创建 Unicode 数据库:

   ```
   CREATE DATABASE UNIDB USING CODESET UTF-8 TERRITORY US
   ```

   其中 UNIDB 是 Unicode 数据库的名称。
4. 编辑 unidb.ddl 脚本并将出现的所有数据库名称更改为新的 Unicode 数据库名称:

   ```
   CONNECT TO UNIDB
   ```

   要保留现有数据库，还必须在 unidb.ddl 文件中更改表空间的文件名规范。否则，可以删除现有数据库并使用相同的表空间文件:

   ```
   DROP DATABASE SAMPLE
   ```
5. 通过运行已编辑的 DDL 脚本来重新创建数据库结构:

   ```
   db2 -tvf unidb.ddl
   ```
6. 使用 **db2move** 命令将数据导入到新的 Unicode 数据库中:

   ```
   cd <export-dir>
   db2move unidb import
   ```

   其中 <export-dir> 是已导出数据的目录，而 UNIDB 是 Unicode 数据库名称。

相关概念:

- 第 333 页的『DB2 数据库 Linux 版、UNIX 版和 Windows 版中的 Unicode 实施』

- 『本机 XML 数据存储概述』（*《XML 指南》*）
- 『XML 数据类型』（*《XML 指南》*）

相关任务:

- 第 337 页的『创建 Unicode 数据库』

相关参考:

- 『db2look - DB2 statistics and DDL extraction tool command』（*Command Reference*）
- 『db2move - Database movement tool command』（*Command Reference*）
- 『DROP DATABASE command』（*Command Reference*）
- 『CONNECT (Type 1) statement』（*SQL Reference, Volume 2*）
- 『CONNECT (Type 2) statement』（*SQL Reference, Volume 2*）

Unicode 文字

可以用下列两种方式指定 Unicode 文字:

- 作为图形字符串常量，使用 G'...' 或 N'....' 格式。用这种方式指定的任何文字将由数据库管理器从应用程序代码页转换为 16 位 Unicode。
- 作为 Unicode 十六进制字符串，使用 UX'....' 或 GX'....' 格式。在大尾数法排序中，在 UX 或 GX 后面的引号内指定的常量必须是 4 个十六进制数字的倍数。每个四位组表示一个 16 位 Unicode 代码点。注意代用字符始终成对出现，因此需要两个四位组来表示高位和低位代用字符。

当使用命令行处理器（CLP） 时，若本地应用程序代码页中存在 UCS-2 字符，则第一种方法比较容易（例如，从使用代码页 850 的终端输入代码页 850 的任何字符）。对于在应用程序代码页指令表之外的字符，应使用第二种方法（例如，从使用代码页 850 的终端指定日语字符）。

相关概念:

- 第 331 页的『Unicode 字符编码』
- 第 333 页的『DB2 数据库 Linux 版、UNIX 版和 Windows 版中的 Unicode 实施』

相关参考:

- 『Constants』（*SQL Reference, Volume 1*）

Unicode 数据库中的字符串比较

模式匹配是现有的 MBCS 数据库的行为与 UCS-2 数据库的行为稍微不同的一个方面。

对于 DB2 数据库 Linux 版、UNIX 版和 Windows 版中的 MBCS 数据库，当前行为如下所示: 若匹配表达式包含 MBCS 数据，则模式可以包含 SBCS 和非 SBCS 字符。该模式中的特殊字符解释如下:

- SBCS 半角下划线字符是一个 SBCS 字符。
- 非 SBCS 半角下划线字符是一个非 SBCS 字符。
- 百分比字符（SBCS 半角或非 SBCS 全角）表示零个或多个 SBCS 或非 SBCS 字符。

在 Unicode 数据库中，“单字节”与“非单字节”字符之间没有任何真正的差别。尽管 UTF-8 格式是 Unicode 字符的“混合字节”编码，但 UTF-8 格式的 SBCS 和非 SBCS 字符之间没有任何真正的区别。每个字符是一个 Unicode 字符，而与采用 UTF-8 格式的字节数无关。在 Unicode 图形列中，每个非补充字符（包括半角下划线字符（U+005F）和半角百分比字符（U+0025））的宽度为两个字节。对于 Unicode 数据库，该模式中的特殊字符解释如下:

- 对于字符串，半角下划线字符（X'5F'）或全角下划线字符（X'EFBCBF'）表示一个 Unicode 字符。半角百分比字符（X'25'）或全角百分比字符（X'EFBC85'）表示零个或多个 Unicode 字符。
- 对于图形字符串，半角下划线字符（U+005F）或全角下划线（U+FF3F）表示一个 Unicode 字符。半角百分比字符（U+0025）或全角百分比字符（U+FF05）表示零个或多个 Unicode 字符。

注: 需要两条下划线才能与一个 Unicode 补充图形字符相匹配，因为这样一个字符在 GRAPHIC 列中由两个 UCS-2 字符表示。只需要一个下划线字符就可以与 CHAR 列中的 Unicode 补充字符相匹配。

对于可选的“转义表达式”，它指定一个用来修改下划线字符和百分比符号字符特殊意义的字符，表达式可以由下列任何一项指定:

- 常量
- 专用寄存器
- 主变量
- 标量函数，其操作数是以上所述的任何内容
- 并置以上任何内容的表达式

带有如下限制:

- 表达式中的元素不能是 LONG VARCHAR、CLOB、LONG VARGRAPHIC 或 DBCLOB。另外，它不能是 BLOB 文件引用变量。
- 对于 CHAR 列，表达式的结果必须是一个字符或者只包含一个字节的二进制字符串（SQLSTATE 22019）。对于 GRAPHIC 列，表达式的结果必须是一个字符（SQLSTATE 22019）。

相关概念:

- 第 331 页的『Unicode 字符编码』
- 第 333 页的『DB2 数据库 Linux 版、UNIX 版和 Windows 版中的 Unicode 实施』

相关参考:

- 『Character strings』（*SQL Reference, Volume 1*）
- 『Graphic strings』（*SQL Reference, Volume 1*）

安装先前的表以便在代码页 1394 与 Unicode 之间进行转换

已经增强了代码页 1394（也称为 Shift JIS X0213）和 Unicode 的转换表。日语 Shift JIS X0213（1394）与 Unicode 之间的转换现在遵从最终 JIS X0213 字符的 ISO/IEC 10646-1:2000 Amendment-1。通过 FTP 可以从网址 `ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/` 获得转换表的先前版本。

过程:

要安装先前定义，以便在 Shift JIS X0213 与 Unicode 之间进行转换:

1. 停止 DB2 数据库 Linux 版、UNIX 版和 Windows 版实例。
2. 将 Web 浏览器指向 `ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/`，或者使用 FTP 连接至 `ftp.software.ibm.com` 站点。此 FTP 服务器是匿名的。
3. 如果正在通过命令行进行连接，请输入 `anonymous` 作为用户标识并输入您的电子邮件地址作为密码来登录。
4. 在登录之后，切换至转换表目录:

   ```
   cd ps/products/db2/info/vr8/conv
   ```
5. 将 1394ucs4.cnv 和 ucs41394.cnv 两个文件以二进制格式复制到 `sqllib/conv/` 目录中。
6. 重新启动 DB2 实例。

相关概念:

- 第 333 页的『DB2 数据库 Linux 版、UNIX 版和 Windows 版中的 Unicode 实施』

相关参考:

- 第 291 页的『受支持的地域代码和代码页』

编码字符集标识（CCSID）943 的备用 Unicode 转换表

日语代码页有几个 IBM 编码字符集标识（CCSID）。CCSID 943 已注册为日语版 Microsoft Windows Shift-JIS 代码页。在 CCSID 943 与 Unicode 之间转换字符时，您可能会遇到下列两个问题。这些问题是 IBM 代码页转换表与 Microsoft 代码页转换表之间的差别而引起的。

问题 1: :

由于历史原因，在 CCSID 943 代码页中，有 300 多个字符可以由两种或三种代码点来表示。在使用输入方法编辑器（IME）和代码页转换表时，将只输入这些等同代码点的其中一个代码点。例如，罗马数字 1 的小写字符（ “i” ）有两个等同的代码点: X’EEEF’ 和 X’FA40’。输入 “i” 时，Microsoft Windows IME 始终生成 X’FA40’。通常，IBM 和 Microsoft 使用同一个主代码点来表示该字符，但下列 13 个字符除外:

表 104. CCSID 943 Shift-JIS 代码点转换

字符名（Unicode 代码点）	IBM Shift-JIS 主代码点	Microsoft Shift-JIS 主代码点
罗马数字 1（U+2160）	X’FA4A’	X’8754’
罗马数字 2（U+2161）	X’FA4B’	X’8755’
罗马数字 3（U+2162）	X’FA4C’	X’8756’
罗马数字 4（U+2163）	X’FA4D’	X’8757’
罗马数字 5（U+2164）	X’FA4E’	X’8758’
罗马数字 6（U+2165）	X’FA4F’	X’8759’
罗马数字 7（U+2166）	X’FA50’	X’875A’
罗马数字 8（U+2167）	X’FA51’	X’875B’
罗马数字 9（U+2168）	X’FA52’	X’875C’

表 104. CCSID 943 Shift-JIS 代码点转换 (续)

字符名（Unicode 代码点）	IBM Shift-JIS 主代码点	Microsoft Shift-JIS 主代码点
罗马数字 10（U+2169）	X’FA53’	X’875D’
带括号有（U+3231）	X’FA58’	X’878A’
数字符（U+2116）	X’FA59’	X’8782’
电话符（U+2121）	X’FA5A’	X’8784’

诸如 DB2 数据库管理器之类的 IBM 产品主要使用 IBM 代码点，例如，使用 X’FA4A’ 来表示大写罗马数字 “I”，但 Microsoft 产品使用 X’8754’ 来表示该字符。Microsoft ODBC 应用程序可以将 “I” 字符作为 X’8754’ 插入到 CCSID 为 943 的 DB2 数据库中，并且 DB2 控制中心可以将同一字符作为 X’FA4A’ 插入到 CCSID 为 943 的同一数据库中。但是，Microsoft ODBC 应用程序只能找到将 “I” 编码为 X’8754’ 的行，而 DB2 控制中心只能找到将 “I” 编码为 X’FA4A’ 的行。要使 DB2 控制中心能够选择将 “I” 编码为 X’8754’，需要将用于从 Unicode 转换到 CCSID 943 的缺省 IBM 转换表替换为 DB2 数据库管理器提供的备用 Microsoft 转换表。

问题 2: :

根据使用的是 IBM 转换表还是 Microsoft 转换表，把下列字符从 CCSID 943 转换为 Unicode 时将产生不同的代码点。对于这些字符来说， IBM 转换表符合日本工业标准 JISX0208、JISX0212 和 JISX0221 中指定的字符名称。

表 105. CCSID 943 到 Unicode 代码点转换

Shift-JIS 代码点（字符名）	IBM 主代码点（Unicode 名）	Microsoft 主代码点（Unicode 名）
X’815C’（EM 破折号）	U+2014（EM 破折号）	U+2015（水平线）
X’8160’（波浪线）	U+301C（波浪线）	U+FF5E（全宽颚音符号）
X’8161’（双竖线）	U+2016（双竖线）	U+2225（平行号）
X’817C’（减号）	U+2212（减号）	U+FF0D（全宽连字符-减号）
X’FA55’（竖杠）	U+00A6（竖杠）	U+FFE4（全宽竖杠）

例如，使用 IBM 转换表时，CCSID 943 代码点为 X’815C’ 的 EM 破折号将转换为 Unicode 代码点 U+2014，但是，在使用 Microsoft 转换表时，它将转换为 U+2015。由于 Microsoft ODBC 应用程序把 U+2014 视为无效代码点，所以对于 Microsoft ODBC 应用程序来说，这会产生潜在问题。为了避免这些潜在问题，需要将用于从 CCSID 943 转换到 Unicode 的缺省 IBM 转换表替换为 DB2 数据库管理器提供的备用 Microsoft 转换表。

只有在闭合环境中才应该在 CCSID 943 与 Unicode 之间使用备用 Microsoft 转换表，在这些环境中，运行 CCSID 943 的 DB2 客户机和 DB2 数据库都使用相同的备用 Microsoft 转换表。如果一个 DB2 客户机正在使用缺省 IBM 转换表，而另一个客户机正在使用备用 Microsoft 转换表，并且这两个客户机都将数据插入到 CCSID 为 943 的相同 DB2 数据库中，则同一个字符在数据库中可能存储为不同代码点。

相关概念:

- 第 331 页的『Unicode 字符编码』

相关任务:

• 第 343 页的『将编码字符集标识（CCSID）943 的 Unicode 转换表替换为 Microsoft 转换表』

将编码字符集标识（CCSID）943 的 Unicode 转换表替换为 Microsoft 转换表

在编码字符集标识（CCSID）943 与 Unicode 之间转换时，将使用 DB2 数据库 Linux 版、UNIX 版和 Windows 版数据库管理器缺省代码页转换表。如果要使用另一版本（例如，Microsoft 版本）的转换表，则必须手工覆盖缺省转换表。

先决条件:

如果 `sqllib` 目录的 `conv` 子目录中已经存在想要覆盖的代码页转换表文件，则应备份该文件以防您想要还原为缺省表。

限制:

要让转换表替换结果生效，必须更改数据库服务器及其所有客户机上的转换表。

过程:

要替换用来在 CCSID 943 与 Unicode 之间进行转换的 DB2 缺省转换表:

1. 在客户机上替换转换表时，停止正在使用数据库的所有应用程序。如果有任何 CLP 会话正在运行，则对每个会话发出 TERMINATE 命令。替换数据库服务器上的转换表时，通过发出 **db2stop** 命令来停止所有节点上的所有实例。
2. 将 `sqllib/conv/ms/0943ucs2.cnv` 复制至 `sqllib/conv/0943ucs2.cnv`。
3. 将 `sqllib/conv/ms/ucs20943.cnv` 复制至 `sqllib/conv/ucs20943.cnv`。
4. 重新启动所有应用程序。

相关概念:

• 第 341 页的『编码字符集标识（CCSID）943 的备用 Unicode 转换表』

编码字符集标识（CCSID）954 的备用 Unicode 转换表

日语代码页有几个 IBM 编码字符集标识（CCSID）。CCSID 954 已注册为日语 EUC 代码页。CCSID 954 是日语 UNIX 和 Linux 平台的公共编码。当使用 Microsoft ODBC 应用程序来连接到使用 CCSID 954 的 DB2 数据库时，在将 CCSID 为 954 的数据转换为 Unicode 时可能会遇到一些潜在问题。这些问题是 IBM 的代码页转换表与 Microsoft 的代码页转换表之间的差别而引起的。

根据使用的转换表（IBM 或 Microsoft）的不同，下列字符在从 CCSID 954 转换到 Unicode 时会产生不同的代码点。对于这些字符来说，IBM 转换表符合日本工业标准（JIS）JISX0208、JISX0212 和 JISX0221 指定的字符名称。

表 106. CCSID 954 到 Unicode 代码点转换

EUC-JP 代码点（字符名）	IBM 主代码点（Unicode 名）	Microsoft 主代码点（Unicode 名）
X’A1BD’（EM 破折号）	U+2014（EM 破折号）	U+2015（水平线）

表 106. CCSID 954 到 Unicode 代码点转换（续）

EUC-JP 代码点（字符名）	IBM 主代码点（Unicode 名）	Microsoft 主代码点（Unicode 名）
X'A1C1'（波浪线）	U+301C（波浪线）	U+FF5E（全宽颚音符号）
X'A1C2'（双竖线）	U+2016（双竖线）	U+2225（平行号）
X'A1DD'（减号）	U+2212（减号）	U+FF0D（全宽连字符-减号）
X'8FA2C3'（竖杠）	U+00A6（竖杠）	U+FFE4（全宽竖杠）

例如，使用 IBM 转换表时，CCSID 954 代码点为 X'A1BD' 的 EM 破折号将转换为 Unicode 代码点 U+2014，但是，在使用 Microsoft 转换表时，它将转换为 U+2015。由于 Microsoft ODBC 应用程序把 U+2014 视为无效代码点，所以对于 Microsoft ODBC 应用程序来说，这会产生潜在问题。为了避免这些潜在问题，需要将用于从 CCSID 954 转换到 Unicode 的缺省 IBM 转换表替换为 DB2 数据库管理器提供的备用 Microsoft 转换表。

相关概念:

- 第 344 页的『将编码字符集标识（CCSID）954 的 Unicode 转换表替换为 Microsoft 转换表』
- 第 331 页的『Unicode 字符编码』

将编码字符集标识（CCSID）954 的 Unicode 转换表替换为 Microsoft 转换表

当从编码字符集标识（CCSID）954 转换到 Unicode 时，将使用 DB2 数据库管理器缺省代码页转换表。如果要使用另一版本（例如，Microsoft 版本）的转换表，则必须手工覆盖缺省转换表。

先决条件: :

如果 *sqllib* 目录的 *conv* 子目录中已经存在想要覆盖的代码页转换表文件，则应备份该文件以防您想要还原为缺省表。

限制: :

要让转换表替换结果生效，连接至同一数据库的每个 DB2 客户机都必须更改其转换表。如果客户端是 ANSI 代码页为 Shift-JIS（CCSID 943）的日语版 Windows，则还需要将 CCSID 943 与 Unicode 之间的缺省转换表更改为 Microsoft 版本。否则，不同的客户机可能会使用不同的代码点来存储同一字符。

过程: :

要替换用于从 CCSID 954 转换到 Unicode 的 DB2 缺省转换表，请执行下列步骤:

1. 在客户机上替换转换表时，停止正在使用数据库的所有应用程序。如果有任何 CLP 会话正在运行，则对每个会话发出 **TERMINATE** 命令。替换数据库服务器上的转换表时，通过发出 **db2stop** 命令来停止所有节点上的所有实例。
2. 将 *sqllib/conv/ms/0954ucs2.cnv* 复制到 *sqllib/conv/0954ucs2.cnv*。
3. 重新启动所有应用程序。

要替换用于在 CCSID 943 与 Unicode 之间进行转换的 DB2 缺省转换表，请执行下列步骤:

1. 在客户机上替换转换表时，停止正在使用数据库的所有应用程序。如果有任何 CLP 会话正在运行，则对每个会话发出 **TERMINATE** 命令。替换数据库服务器上的转换表时，通过发出 **db2stop** 命令来停止所有节点上的所有实例。
2. 将 *sqllib/conv/ms/0943ucs2.cnv* 复制至 *sqllib/conv/0943ucs2.cnv*。
3. 将 *sqllib/conv/ms/ucs20943.cnv* 复制至 *sqllib/conv/ucs20943.cnv*。
4. 重新启动所有应用程序。

相关概念:

- 第 343 页的『编码字符集标识（CCSID）954 的备用 Unicode 转换表』
- 第 331 页的『Unicode 字符编码』

编码字符集标识（CCSID）5026 的备用 Unicode 转换表

日语代码页有几个 IBM 编码字符集标识（CCSID）。CCSID 5026 已注册为日语 EBCDIC 代码页。使用 Microsoft ODBC 应用程序连接至 CCSID 为 5026 的 DB2 主机数据库时，在将 CCSID 为 5026 的数据转换为 Unicode 时可能会遇到一些潜在问题。这些问题是 IBM 的代码页转换表与 Microsoft 的代码页转换表之间的差别而引起的。根据使用的转换表（IBM 或 Microsoft）的不同，下列字符在从 CCSID 5026 转换到 Unicode 时会产生不同的代码点。对于这些字符来说，IBM 转换表符合日本工业标准（JIS）JISX0208、JISX0212 和 JISX0221 指定的字符名称。

表 107. CCSID 5026 到 Unicode 代码点转换

EBCDIC 代码点（字符名）	IBM 主代码点（Unicode 名）	Microsoft 主代码点（Unicode 名）
X’444A’（EM 破折号）	U+2014（EM 破折号）	U+2015（水平线）
X’43A1’（波浪线）	U+301C（波浪线）	U+FF5E（全宽颚音符号）
X’447C’（双竖线）	U+2016（双竖线）	U+2225（平行号）
X’4260’（减号）	U+2212（减号）	U+FF0D（全宽连字符-减号）
X’426A’（竖杠）	U+00A6（竖杠）	U+FFE4（全宽竖杠）

例如，使用 IBM 转换表时，CCSID 5026 代码点为 X’444A’ 的 EM 破折号将转换为 Unicode 代码点 U+2014，但是，在使用 Microsoft 转换表时，它将转换为 U+2015。由于 Microsoft ODBC 应用程序把 U+2014 视为无效代码点，所以对于 Microsoft ODBC 应用程序来说，这会产生潜在问题。为了避免这些潜在问题，需要将用于从 CCSID 5026 转换到 Unicode 的缺省 IBM 转换表替换为 DB2 数据库管理器提供的备用 Microsoft 转换表。

相关概念:

- 第 346 页的『将编码字符集标识（CCSID）的 Unicode 转换表替换为 Microsoft 转换表』
- 第 331 页的『Unicode 字符编码』

将编码字符集标识（CCSID）的 Unicode 转换表替换为 Microsoft 转换表

当从编码字符集标识（CCSID）5026 转换到 Unicode 时，将使用 DB2 数据库管理器缺省代码页转换表。如果要使用另一版本（例如，Microsoft 版本）的转换表，则必须手工覆盖缺省转换表。

先决条件: :

如果 *sqllib* 目录的 *conv* 子目录中已经存在想要覆盖的代码页转换表文件，则应备份该文件以防您想要还原为缺省表。

限制: :

要让转换表替换结果生效，连接至同一数据库的每个 DB2 客户机都必须更改其转换表。

这个 Microsoft 转换表仅用于以 CCSID 5026 或 930 编码的数据，而不能用于以 CCSID 1390 编码的数据。由于 DB2 数据库管理器对以 CCSID 5026、930 和 1390 编码的数据使用同一个转换表，这意味着一旦将缺省 IBM 转换表替换为 Microsoft 转换表，就不应该选择任何以 CCSID 1390 编码的数据。

激活此备用 Microsoft 转换表不会将以 5026 编码的图形数据的代码页转换行为更改为 Unicode。要使用备用 Microsoft 转换表将以 5026 编码的图形数据转换为 Unicode，除了下面概述的过程之外，还必须将文件 *sqllib/conv/ms/0939ucs2.cnv* 复制至 *sqllib/conv/1399ucs2.cnv*。完成这些步骤之后，从以下 CCSID 到 Unicode 的字符数据和图形数据转换也将使用这个 Microsoft 转换表: 5026、930、1390、5035、939 和 1399。

过程: :

要替换用于从 CCSID 5026 转换到 Unicode 的 DB2 缺省转换表，请执行下列步骤:

1. 在客户机上替换转换表时，停止正在使用数据库的所有应用程序。如果有任何 CLP 会话正在运行，则对每个会话发出 *db2 terminate* 命令。
2. 将 *sqllib/conv/ms/0930ucs2.cnv* 复制到 *sqllib/conv/1390ucs2.cnv*。
3. 重新启动所有应用程序。

相关概念:

- 第 345 页的『编码字符集标识（CCSID）5026 的备用 Unicode 转换表』

编码字符集标识（CCSID）5035 的备用 Unicode 转换表

日语代码页有几个 IBM 编码字符集标识（CCSID）。CCSID 5035 已注册为日语 EBCDIC 代码页。使用 Microsoft ODBC 应用程序连接至 CCSID 为 5035 的 DB2 主机数据库时，在将 CCSID 为 5035 的数据转换为 Unicode 时可能会遇到一些潜在问题。这些问题是 IBM 的代码页转换表与 Microsoft 的代码页转换表之间的差别而引起的。根据使用的转换表（IBM 或 Microsoft）的不同，下列字符在从 CCSID 5035 转换到 Unicode 时会产生不同的代码点。对于这些字符来说，IBM 转换表符合日本工业标准（JIS）JISX0208、JISX0212 和 JISX0221 指定的字符名称。

表 108. CCSID 5035 到 Unicode 代码点转换

EBCDIC 代码点（字符名）	IBM 主代码点（Unicode 名）	Microsoft 主代码点（Unicode 名）
X'444A'（EM 破折号）	U+2014（EM 破折号）	U+2015（水平线）
X'43A1'（波浪线）	U+301C（波浪线）	U+FF5E（全宽颚音符号）
X'447C'（双竖线）	U+2016（双竖线）	U+2225（平行号）
X'4260'（减号）	U+2212（减号）	U+FF0D（全宽连字符-减号）
X'426A'（竖杠）	U+00A6（竖杠）	U+FFE4（全宽竖杠）

例如，使用 IBM 转换表时，CCSID 5035 代码点为 X'444A' 的 EM 破折号将转换为 Unicode 代码点 U+2014，但是，在使用 Microsoft 转换表时，它将转换为 U+2015。由于 Microsoft ODBC 应用程序把 U+2014 视为无效代码点，所以对于 Microsoft ODBC 应用程序来说，这会产生潜在问题。为了避免这些潜在问题，需要将用于从 CCSID 5035 转换到 Unicode 的缺省 IBM 转换表替换为 DB2 数据库管理器提供的备用 Microsoft 转换表。

相关概念:

- 第 331 页的『Unicode 字符编码』
- 第 347 页的『将编码字符集标识（CCSID）5035 的 Unicode 转换表替换为 Microsoft 转换表』

将编码字符集标识（CCSID）5035 的 Unicode 转换表替换为 Microsoft 转换表

当从编码字符集标识（CCSID）5035 转换到 Unicode 时，将使用 DB2 数据库管理器缺省代码页转换表。如果要使用另一版本（例如，Microsoft 版本）的转换表，则必须手工覆盖缺省转换表。

先决条件: :

如果 *sqllib* 目录的 *conv* 子目录中已经存在想要覆盖的代码页转换表文件，则应备份该文件以防您想要还原为缺省表。

限制: :

要让转换表替换结果生效，连接至同一数据库的每个 DB2 客户机都必须更改其转换表。

这个 Microsoft 转换表仅用于以 CCSID 5039 或 939 编码的数据，而不能用于以 CCSID 1399 编码的数据。由于 DB2 数据库管理器对以 CCSID 5035、939 和 1399 编码的数据使用同一个转换表，这意味着一旦将缺省 IBM 转换表替换为 Microsoft 转换表，就不应该选择任何以 CCSID 1399 编码的数据。

一旦将缺省 IBM 转换表替换为 Microsoft 转换表，则从以下 CCSID 到 Unicode 的图形数据转换也将使用这个 Microsoft 转换表：930、1390、939 和 1399。

过程: :

要替换用于从 CCSID 5035 转换到 Unicode 的 DB2 缺省转换表，请执行下列步骤:

1. 在客户机上替换转换表时，停止正在使用数据库的所有应用程序。如果有任何 CLP 会话正在运行，则对每个会话发出 **TERMINATE** 命令。
2. 将 *sqllib/conv/ms/0939ucs2.cnv* 复制到 *sqllib/conv/1399ucs2.cnv*。
3. 重新启动所有应用程序。

相关概念:

- 第 346 页的『编码字符集标识（CCSID）5035 的备用 Unicode 转换表』
- 第 331 页的『Unicode 字符编码』

编码字符集标识（CCSID）5039 的备用 Unicode 转换表

日语代码页有几个 IBM 编码字符集标识（CCSID）。CCSID 943 已注册为日语版 Microsoft Windows Shift-JIS 代码页。但是，在 HP-UX 平台上，Shift-JIS 代码页已注册为 CCSID 5039。CCSID 5039 仅包含日本工业标准（JIS）字符，未包含任何由供应商定义的字符。使用 Microsoft ODBC 应用程序时，在将 CCSID 为 5039 的数据转换为 Unicode 时可能会遇到一些潜在问题。这些问题是 IBM 的代码页转换表与 Microsoft 的代码页转换表之间的差别而引起的。

根据使用的转换表（IBM 或 Microsoft）的不同，下列字符在从 CCSID 5039 转换到 Unicode 时会产生不同的代码点。对于这些字符来说，IBM 转换表符合日本工业标准（JIS）JISX0208 和 JISX0221 指定的字符名称。

表 109. CCSID 5039 到 Unicode 代码点转换

Shift-JIS 代码点（字符名）	IBM 主代码点（Unicode 名）	Microsoft 主代码点（Unicode 名）
X'815C'（EM 破折号）	U+2014（EM 破折号）	U+2015（水平线）
X'8160'（波浪线）	U+301C（波浪线）	U+FF5E（全宽颚音符号）
X'8161'（双竖线）	U+2016（双竖线）	U+2225（平行号）
X'817C'（减号）	U+2212（减号）	U+FF0D（全宽连字符-减号）

例如，使用 IBM 转换表时，CCSID 5039 代码点为 X'815C' 的 EM 破折号将转换为 Unicode 代码点 U+2014，但是，在使用 Microsoft 转换表时，它将转换为 U+2015。由于 Microsoft ODBC 应用程序把 U+2014 视为无效代码点，所以对于 Microsoft ODBC 应用程序来说，这会产生潜在问题。为了避免这些潜在问题，需要将用于从 CCSID 5039 转换到 Unicode 的缺省 IBM 转换表替换为 DB2 数据库管理器提供的备用 Microsoft 转换表。

相关概念:

- 第 349 页的『将编码字符集标识（CCSID）5039 的 Unicode 转换表替换为 Microsoft 转换表』
- 第 331 页的『Unicode 字符编码』

将编码字符集标识（CCSID）5039 的 Unicode 转换表替换为 Microsoft 转换表

当从编码字符集标识（CCSID）5039 转换到 Unicode 时，将使用 DB2 数据库管理器缺省代码页转换表。如果要使用另一版本（例如，Microsoft 版本）的转换表，则必须手工覆盖转换表。

先决条件: :

如果 *sqllib* 目录的 *conv* 子目录中已经存在想要覆盖的代码页转换表文件，则应备份该文件以防您想要还原为缺省表。

限制: :

要让转换表替换结果生效，连接至同一数据库的每个 DB2 客户机都必须更改其转换表。

过程: :

要替换用于从 CCSID 5039 转换到 Unicode 的 DB2 缺省转换表，请执行下列步骤:

1. 在客户机上替换转换表时，停止正在使用数据库的所有应用程序。如果有任何 CLP 会话正在运行，则对每个会话发出 **TERMINATE** 命令。
2. 将 *sqllib/conv/ms/5039ucs2.cnv* 复制到 *sqllib/conv/5039ucs2.cnv*。
3. 重新启动所有应用程序。

相关概念:

- 第 348 页的『编码字符集标识（CCSID）5039 的备用 Unicode 转换表』
- 第 331 页的『Unicode 字符编码』

附录 C. DB2 数据库技术信息

DB2 技术信息概述

DB2 技术信息可通过下列工具和方法获得:

- DB2 信息中心
  - 主题
  - DB2 工具的帮助
  - 样本程序
  - 教程
- DB2 书籍
  - PDF 文件(可下载)
  - PDF 文件(在 DB2 PDF CD 中)
  - 印刷版书籍
- 命令行帮助
  - 命令帮助
  - 消息帮助
- 样本程序

IBM 定期提供文档更新。如果访问 ibm.com® 的 DB2 信息中心上的联机版本,则不必安装文档更新,这是因为 IBM 会维护此版本的更新。如果已经安装了 DB2 信息中心,则建议安装文档更新。IBM 提供新信息时,文档更新允许更新从 *DB2 信息中心 CD* 安装或从 Passport Advantage 下载的信息。

注: DB2 信息中心主题的更新频率比 PDF 书籍或硬拷贝书籍的更新频率高。要获取最新信息,安装可用的文档更新,或者参阅 ibm.com 上的 DB2 信息中心。

可以在线访问 ibm.com 上的其他 DB2 技术信息,如技术说明、白皮书和 Redbooks™。访问位于以下网址的 DB2 信息管理软件库站点:http://www.ibm.com/software/data/sw-library/。

文档反馈

我们非常重视您对 DB2 文档的反馈。如果您想就如何改善 DB2 文档提出建议,请将电子邮件发送至 db2docs@ca.ibm.com。DB2 文档小组会阅读您的所有反馈,但不能直接答复您。请尽可能提供具体的示例,这样我们才能更好地了解您所关心的问题。如果您要提供有关具体主题或帮助文件的反馈,请加上标题和 URL。

请不要用以上电子邮件地址与 DB2 客户支持机构联系。如果您遇到文档不能解决的 DB2 技术问题,请与您当地的 IBM 服务中心联系以获得帮助。

相关概念:

- 『DB2 信息中心的功能部件』(*DB2 在线信息中心*)
- 『Sample files』(*样本主题*)

相关任务:

- 『Invoking command help from the command line processor』(*Command Reference*)
- 『Invoking message help from the command line processor』(*Command Reference*)
- 第 356 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』

相关参考:

- 第 352 页的『PDF 格式的 DB2 技术资料库』

PDF 格式的 DB2 技术资料库

下列各表描述 IBM 出版物中心(网址为 网址为 www.ibm.com/shop/publications/order)提供的 DB2 库。

尽管这些表标识书籍有印刷版,但可能未在您所在国家或地区提供它们。

这些书籍中的信息对于所有 DB2 用户来说都是基础知识;不管您是程序员、数据库管理员还是使用 DB2 Connect 或其他 DB2 产品的人员,都将发现此信息很有用。

表 110. DB2 技术信息

书名	书号	是否提供印刷版
《管理指南:实施》	s151-0278	是
《管理指南:计划》	s151-0280	是
Administrative API Reference	SC10-4231	是
Administrative SQL Routines and Views	SC10-4293	否
Call Level Interface Guide and Reference, Volume 1	SC10-4224	是
Call Level Interface Guide and Reference, Volume 2	SC10-4225	是
Command Reference	SC10-4226	否
《数据移动实用程序指南和参考》	s151-0284	是
《数据恢复和高可用性指南与参考》	s151-0281	是
Developing ADO.NET and OLE DB Applications	SC10-4230	是
Developing Embedded SQL Applications	SC10-4232	是
Developing SQL and External Routines	SC10-4373	否
Developing Java Applications	SC10-4233	是
Developing Perl and PHP Applications	SC10-4234	否
Getting Started with Database Application Development	SC10-4252	是

表 110. DB2 技术信息 (续)

书名	书号	是否提供印刷版
《Linux 和 Windows 上的 DB2 安装和管理入门》	g151-0300	是
《消息参考：第 1 卷》	s151-0305	否
《消息参考：第 2 卷》	s151-0306	否
《迁移指南》	G151-0481	是
《Net Search Extender 管理和用户指南》 **注：** 此文档的 HTML 不是从 HTML 文档 CD 安装的。	S151-0360	是
《性能指南》	s151-0279	是
Query Patroller Administration and User's Guide	GC10-4241	是
《DB2 客户机快速入门》	g151-0302	否
《DB2 服务器快速入门》	g151-0299	是
Spatial Extender and Geodetic Data Management Feature User's Guide and Reference	SC18-9749	是
SQL Guide	SC10-4248	是
SQL Reference, Volume 1	SC10-4249	是
SQL Reference, Volume 2	SC10-4250	是
《系统监视器指南和参考》	s151-0283	是
《故障诊断指南》	G151-0285	否
《Visual Explain 教程》	S151-0455	否
《新增内容》	s151-0307	是
XML Extender Administration and Programming	SC18-9750	是
《XML 指南》	s151-0282	是
XQuery Reference	SC18-9796	是

表 111. 特定于 DB2 Connect 的技术信息

书名	书号	是否提供印刷版
《DB2 Connect 用户指南》	s151-0304	是
《DB2 Connect 个人版快速入门》	G151-0431	是
《DB2 Connect 服务器快速入门》	g151-0303	是

表 112. WebSphere 信息集成技术信息

书名	书号	是否提供印刷版
WebSphere Information Integration: Administration Guide for Federated Systems	SC19-1020	是

表 112. WebSphere 信息集成技术信息 (续)

书名	书号	是否提供印刷版
WebSphere Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018	是
WebSphere Information Integration: Configuration Guide for Federated Data Sources	SC19-1034	否
WebSphere Information Integration: SQL Replication Guide and Reference	SC19-1030	是

注: DB2 发行说明提供特定于产品的发行版和修订包级别的附加信息。有关更多信息，请参阅相关链接。

相关概念:

- 第 351 页的『DB2 技术信息概述』
- 『关于发行说明』(*《发行说明》*)

相关任务:

- 第 354 页的『订购印刷版 DB2 书籍』

订购印刷版 DB2 书籍

如果您需要印刷版的 DB2 书籍，可以在许多（但不是所有）国家或地区在线购买。无论何时都可以从当地的 IBM 代表处订购印刷版的 DB2 书籍。注意，*DB2 PDF 文档* CD 上的某些软拷贝书籍没有印刷版。例如，*DB2 消息参考*的任何一卷都没有提供印刷版书籍。

只要支付一定费用，就可以从 IBM 获取 DB2 PDF 文档 CD，该 CD 包含许多 DB2 书籍的印刷版。根据您下订单的位置，您可能能够从 IBM 出版物中心在线订购书籍。如果在线订购在您所在国家或地区不可用，您总是可以从当地的 IBM 代表处订购印刷版 DB2 书籍。注意，并非 DB2 PDF 文档 CD 上的所有书籍都有印刷版。

注: 最新最完整的 DB2 文档保留在网址如下的 DB2 信息中心中: http://publib.boulder.ibm.com/infocenter/db2help/。

过程:

要订购印刷版的 DB2 书籍:

- 要了解您是否可从所在国家或地区在线订购印刷版的 DB2 书籍，可查看 IBM 出版物中心站点，网址为: http://www.ibm.com/shop/publications/order。 必须先选择国家、地区或语言才能访问出版物订购信息，然后再按照针对您所在位置的订购指示信息进行订购。
- 要从当地的 IBM 代表处订购印刷版的 DB2 书籍:
  – 从下列其中一个 Web 站点找到当地代表处的联系信息:
    - IBM 全球联系人目录，网址为 www.ibm.com/planetwide

- IBM 出版物 Web 站点，网址为 http://www.ibm.com/shop/publications/order。必须先选择国家、地区或语言才能访问对应您的所在地的出版物主页。在此页面中访问“关于此站点”链接。
- 请在致电时说明您想订购 DB2 出版物。
- 请向您当地的代表提供想要订购的书籍的书名和书号。

相关概念:

- 第 351 页的『DB2 技术信息概述』

相关参考:

- 第 352 页的『PDF 格式的 DB2 技术资料库』

从命令行处理器显示 SQL 状态帮助

DB2 返回可作为 SQL 语句结果的条件的 SQLSTATE 值。SQLSTATE 帮助说明 SQL 状态和 SQL 状态类代码的含义。

过程:

要调用 SQL 状态帮助，打开命令行处理器并输入:

`?` *`sqlstate`* 或 `?` *`class code`*

其中，*sqlstate* 表示有效的 5 位 SQL 状态，*class code* 表示该 SQL 状态的前 2 位。

例如，`? 08003` 显示 08003 SQL 状态的帮助，而 `? 08` 显示 08 类代码的帮助。

相关任务:

- 『Invoking command help from the command line processor』（*Command Reference*）
- 『Invoking message help from the command line processor』（*Command Reference*）

访问不同版本的 DB2 信息中心

对于 DB2 版本 9 主题，DB2 信息中心 URL 为 http://publib.boulder.ibm.com/infocenter/db2luw/v9/。

对于 DB2 版本 8 主题，请访问以下版本 8 信息中心 URL: http://publib.boulder.ibm.com/infocenter/db2luw/v8/。

相关任务:

- 第 356 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』

以首选语言显示 DB2 信息中心中的主题

DB2 信息中心尝试以您在浏览器首选项中指定的语言显示主题。如果未提供主题的首选语言翻译版本，则 DB2 信息中心将显示该主题的英文版。

过程:

要在 Internet Explorer 浏览器中以您的首选语言显示主题:

1. 在 Internet Explorer 中，单击**工具** —> **Internet 选项** —> **语言...**按钮。“语言首选项”窗口打开。
2. 确保您的首选语言被指定为语言列表中的第一个条目。
   - 要将新语言添加至列表，单击**添加...** 按钮。

     注: 添加语言并不能保证计算机具有以首选语言显示主题所需的字体。
   - 要将语言移至列表顶部，选择该语言并单击**上移**按钮直到该语言成为语言列表中的第一个条目。
3. 清除浏览器高速缓存然后刷新页面以便以首选语言显示 DB2 信息中心。

要在 Firefox 或 Mozilla 浏览器中以首选语言显示主题:
1. 选择**工具** —> **选项** —> **语言**按钮。“语言”面板将显示在“首选项”窗口中。
2. 确保您的首选语言被指定为语言列表中的第一个条目。
   - 要将新语言添加至列表，单击**添加...** 按钮以从“添加语言”窗口中选择一种语言。
   - 要将语言移至列表顶部，选择该语言并单击**上移**按钮直到该语言成为语言列表中的第一个条目。
3. 清除浏览器高速缓存然后刷新页面以便以首选语言显示 DB2 信息中心。

在某些浏览器和操作系统组合上，可能还必须将操作系统的区域设置更改为您选择的语言环境和语言。

相关概念:
- 第 351 页的『DB2 技术信息概述』

更新安装在计算机或内部网服务器上的 DB2 信息中心

如果在本地安装了 DB2 信息中心，则可以下载已更新的主题。大多数主题底部的“最近一次更新日期”值指示该主题的当前级别。

要确定整个 DB2 信息中心是否存在更新，请查找“信息中心”主页上的“最近一次更新日期”值。将本地安装的主页中的值与 http://www.ibm.com/software/data/db2/udb/support/icupdate.html 上最新可下载更新的日期进行比较。如果提供了较新的可下载更新，就可以更新本地安装的信息中心。

更新在本地安装的 DB2 信息中心要求您:
1. 停止计算机上的 DB2 信息中心，然后以独立方式重新启动信息中心。如果以独立方式运行信息中心，则网络上的其他用户将无法访问信息中心，因而您可以下载和应用更新。
2. 使用更新功能来确定 IBM 是否提供了更新包。

   注: 在 CD 上也提供了更新。有关如何配置信息中心以从 CD 安装更新的详细信息，请参阅相关链接。

   如果提供了更新包，则使用更新功能来下载这些更新包。(更新功能只能用于独立方式。)
3. 停止独立信息中心，然后在计算机上重新启动 DB2 信息中心服务。

过程:

要更新安装在您的计算机或内部网服务器上的 DB2 信息中心:

1. 停止 DB2 信息中心服务。
   - 在 Windows 上，单击**开始** → **控制面板** → **管理工具** → **服务**。然后右键单击 **DB2 信息中心**服务，并选择**停止**。
   - 在 Linux 上，输入以下命令:
     ```
     /etc/init.d/db2icdv9 stop
     ```
2. 以独立方式启动信息中心。
   - 在 Windows 上:
     a. 打开命令窗口。
     b. 浏览至信息中心的安装路径。在缺省情况下，DB2 信息中心安装在 `C:\Program Files\IBM\DB2 Information Center\Version 9` 目录中。
     c. 使用 DB2 信息中心的标准路径运行 `help_start.bat` 文件:
        ```
        <DB2 Information Center dir>\doc\bin\help_start.bat
        ```
   - 在 Linux 上:
     a. 浏览至信息中心的安装路径。在缺省情况下，DB2 信息中心安装在 `/opt/ibm/db2ic/V9` 目录中。
     b. 使用 DB2 信息中心的标准路径运行 `help_start` 脚本:
        ```
        <DB2 Information Center dir>/doc/bin/help_start
        ```

   系统缺省 Web 浏览器将启动以显示独立信息中心。
3. 单击“更新”按钮（ ）。在信息中心的右边面板上，单击**查找更新**。将显示现有文档的更新列表。
4. 要启动下载进程，请检查您想要下载的选项，然后单击**安装更新**。
5. 在完成下载和安装进程后，单击**完成**。
6. 停止独立信息中心。
   - 在 Windows 上，使用 DB2 信息中心的标准路径运行 `help_end.bat` 文件:
     ```
     <DB2 Information Center dir>\doc\bin\help_end.bat
     ```

     注: help_end 批处理文件包含安全终止用 help_start 批处理文件启动的进程所需的命令。不要使用 `Ctrl-C` 或任何其他方法来终止 `help_start.bat`。
   - 在 Linux 上，使用 DB2 信息中心的标准路径运行 `help_end` 脚本:
     ```
     <DB2 Information Center dir>/doc/bin/help_end
     ```

     注: help_end 脚本包含安全终止用 help_start 脚本启动的进程所需的命令。不要使用任何其他方法来终止 `help_start` 脚本。
7. 重新启动 DB2 信息中心服务。
   - 在 Windows 上，单击**开始** → **控制面板** → **管理工具** → **服务**。然后右键单击 **DB2 信息中心**服务，并选择**启动**。
   - 在 Linux 上，输入以下命令:
     ```
     /etc/init.d/db2icdv9 start
     ```

更新后的 DB2 信息中心将显示新的主题和更新后的主题。

相关概念:

- 『DB2 信息中心安装选项』（《*DB2 服务器快速入门*》）

相关任务:

- 『使用 DB2 安装向导安装 DB2 信息中心(Linux)』(*《DB2 服务器快速入门》*)
- 『使用 DB2 安装向导安装 DB2 信息中心(Windows)』(*《DB2 服务器快速入门》*)

DB2 教程

DB2 教程帮助您了解 DB2 产品的各个方面。这些课程提供了逐步指示信息。

开始之前:

可从信息中心查看 XHTML 版的教程: http://publib.boulder.ibm.com/infocenter/db2help/ 。

某些课程使用了样本数据或代码。有关其特定任务的任何先决条件的描述,请参阅教程。

DB2 教程:

要查看教程,单击标题。

本机 XML 数据存储
: 设置 DB2 数据库以存储 XML 数据以及如何对本机 XML 数据存储执行基本操作。

《Visual Explain 教程》
: 使用 Visual Explain 来分析、优化和调整 SQL 语句以获取更好的性能。

相关概念:

- 『Visual Explain 概述』(*《管理指南: 实施》*)

DB2 故障诊断信息

有大量故障诊断和问题确定信息可帮助您使用 DB2 产品。

DB2 文档
: 故障诊断信息可在 DB2 信息中心的"DB2 故障诊断指南"或"支持和故障诊断"部分找到。可在该处找到有关如何使用 DB2 诊断工具和实用程序隔离和找出问题的信息、某些最常见问题的解决方案以及有关如何解决使用 DB2 产品时可能遇到的问题的建议。

DB2 技术支持 Web 站点
: 如果您遇到了问题并且想要获取查找可能的原因和解决方案的帮助,请参阅 DB2 技术支持 Web 站点。该"技术支持"站点具有指向最新 DB2 出版物、技术说明、授权程序分析报告(APAR 或错误修订)、修订包和其他资源的链接。可搜索此知识库并查找问题的可能解决方案。

  访问位于以下网址的 DB2 技术支持 Web 站点: http://www.ibm.com/software/data/db2/udb/support.html

相关概念:

- 『问题确定简介』(*《故障诊断指南》*)

• 第 351 页的『DB2 技术信息概述』

条款和条件

如果符合以下条款和条件，则授予您使用这些出版物的准用权。

个人使用: 只要保留所有的专有权声明，您就可以为个人、非商业使用复制这些出版物。未经 IBM 明确同意，您不可以分发、展示或制作这些出版物或其中任何部分的演绎作品。

商业使用: 只要保留所有的专有权声明，您就可以仅在企业内复制、分发和展示这些出版物。未经 IBM 明确同意，您不可以制作这些出版物的演绎作品，或者在您的企业外部复制、分发或展示这些出版物或其中的任何部分。

除非本准用权中有明确授权，不得把其他准用权、许可或权利（无论是明示的还是暗含的）授予其中包含的出版物或任何信息、数据、软件或其他知识产权。

当使用这些出版物损害了 IBM 的利益，或者根据 IBM 的规定，未正确遵守上述指导说明时，则 IBM 保留自主决定撤销本文授予的准用权的权利。

您不可以下载、出口或再出口本信息，除非完全遵守所有适用的法律和法规，包括所有美国出口法律和法规。

IBM 对这些出版物的内容不作任何保证。这些出版物“按现状”提供，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的关于适销和适用于某种特定用途的保证。

附录 D. 声明

IBM 可能不在所有国家或地区提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation"按现状"提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本资料中描述的产品和／或程序进行改进和／或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：（i）允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario

L6G 1C7
CANADA

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息可能包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名称都是虚构的，与实际商业企业所用的名称和地址的任何雷同纯属巧合。

版权许可:

本信息可能包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。

凡这些样本程序的每份拷贝或其任何部分或任何衍生产品，都必须包括如下版权声明:

©（*贵公司的名称*）（*年*）。此部分代码是根据 IBM 公司的样本程序衍生出来的。© Copyright IBM Corp. *（输入年份）*. All rights reserved.

商标

DB2 版本 9 文档库的各个文档中标识的公司、产品或服务名称可能是 International Business Machines Corporation 或其他公司的商标或服务标记。有关 IBM Corporation 在美国和／或其他国家的商标的信息在 http://www.ibm.com/legal/copytrade.shtml 中。

下列各项是其他公司的商标或注册商标，且已在 DB2 文档库中的至少一份文档中使用:

Microsoft、Windows、Windows NT® 和 Windows 徽标是 Microsoft Corporation 在美国和／或其他国家或地区的商标。

Intel、Itanium®、Pentium® 和 Xeon® 是 Intel Corporation 在美国和／或其他国家或地区的商标。

Java 和所有基于 Java 的商标是 Sun Microsystems, Inc. 在美国和／或其他国家或地区的商标。

UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。

Linux 是 Linus Torvalds 在美国和／或其他国家或地区的商标。

其他公司、产品或服务名称可能是其他公司的商标或服务标记。

索引

[A]

[B]

[C]

[D]

[F]

[O]

[P]

[Q]

[R]

[S]

[T]

[W]

[X]

[Y]

[Z]

[特别字符]

A

B

C

D

E

I

L

M

N

O

R

S

T

U

V

W

X

与 IBM 联系

要与您所在国家或地区的 IBM 联系，请查看网址如下的 IBM 全球联系人目录: http://www.ibm.com/planetwide

要了解有关 DB2 产品的更多信息，请访问 http://www.ibm.com/software/data/db2/。

IBM

中国印刷

s151-0280-00



Spine information:

IBM DB2 DB2 版本 9 管理指南：计划